

❑ Tema 3: Representación del conocimiento e inferencia

❑ 3.5: Representaciones estructuradas

- ❑ Marcos
- ❑ Dependencias conceptuales
- ❑ Guiones

Técnicas de representación

❑ Representaciones básicas

- ❑ Lógica de predicados. Representación en Prolog
- ❑ Redes semánticas
- ❑ Sistemas de producción

❑ Representaciones estructuradas

- ❑ Estructuras de ranura y relleno (*slot & filler*)
 - ❑ Débiles: Marcos (*frames*)
 - ❑ Fuertes
 - ❑ Dependencias Conceptuales (*DCs*)
 - ❑ Guiones (*scripts*)


❑ Estudio comparativo de las técnicas de representación

❑ Lenguajes de representación del conocimiento

Técnicas básicas versus estructuradas

- ❑ Representaciones estructuradas: evolución natural de las básicas
- ❑ Las estructuras de **ranura y relleno** (*slot & filler*) permiten definir conceptos mediante **pares atributo-valor** (*propiedades*)
 - ❑ La ranura podría estar vacía para introducir el relleno posteriormente (*o no*). Pueden imponerse restricciones sobre los posibles rellenos
- ❑ Estructuras de ranura y relleno **débiles versus fuertes**
 - ❑ Depende de la cantidad de conocimiento específico del dominio
 - ❑ **Débil**: poco conocimiento, muy generalista → aplicable a muchos dominios aunque se deben programar ciertas operaciones
 - ❑ Las decisiones sobre qué tipos de objetos y qué relaciones utilizar para representar el conocimiento se dejan al diseñador
 - ❑ **Fuerte**: mucho conocimiento específico → aplicable sólo a ciertos dominios
 - ❑ Las estructuras fuertes añaden conocimiento específico sobre los tipos de objetos y relaciones permitidos

Marcos (*frames*) [Minsky, 1975]

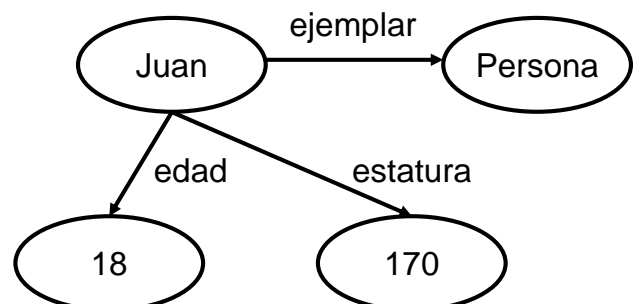
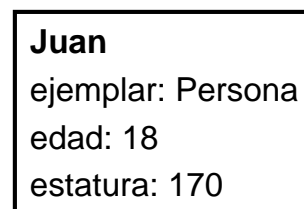
- ❑ Lógica de predicados
 - ❑ Conocimiento factual (terminológico y asertivo)
 - ❑ Orientado a relaciones
 - ❑ Redes semánticas
 - ❑ Conocimiento factual
 - ❑ Orientado a conceptos
 - ❑ Sistemas de producción
 - ❑ Conocimiento procedimental
 - ❑ Marcos
 - ❑ Conocimiento factual + cierto tipo de conocimiento procedimental
 - ❑ Orientado a conceptos
 - ❑ Una entidad o concepto se describe a través de un conjunto de pares atributo/valor (con posibles restricciones para los valores)
 - ❑ Son estructuras de ranura/relleno débiles
- 
- Asignar más estructura a los nodos y a las conexiones

Marcos

- ❑ Idóneos para la organización de una gran cantidad de hechos
 - ❑ Sirven para representar particiones sobre el conjunto de hechos
 - ❑ Lo normal es agrupar hechos que hacen referencia al mismo objeto
 - ❑ Permiten asociar conocimiento procedural relevante a un hecho o a un grupo de hechos
- ❑ Un marco es una colección de atributos y valores
 - ❑ Pretende describir o representar un determinado concepto o un conjunto de conceptos
- ❑ Sistemas de marcos
 - ❑ El empleo de marcos aislados no es usual: suelen construirse sistemas de marcos conectados entre sí
 - ❑ Se razona sobre clases de objetos usando representación del conocimiento prototípico (*cierto en la mayoría de los casos*)
 - ❑ Posibilidad de cambiarlo en las instancias (*excepciones*)

Marcos instancia

- ❑ Entidad individual
 - ❑ Marco de un individuo con 3 pares atributo/valor (estructuras de ranura/relleno)
 - ❑ El relleno de una ranura puede ser un enlace a otro marco
 - ❑ *Persona* es otro marco con sus propias características
 - ❑ 18 y 170 son valores
- ❑ Representación en forma de red semántica
 - ❑ No hay diferencia entre individuos y clases
 - ❑ la representación de *Juan* (individuo) y
 - ❑ la representación de *Persona* (clase de individuos)



Marcos clase

☐ Completitud descriptiva

- ☐ Hechos esenciales que describen a un objeto prototípico de una clase
- ☐ Nos tenemos que asegurar de que siempre aparecen estos hechos en la descripción de un objeto de esa clase
- ☐ Queremos que esa descripción se aplique a un conjunto de objetos

Barco

nombre:

número de identificación:

tipo de barco:

nacionalidad:

tonelaje:

lugar:

☐ Marco clase

- ☐ Generaliza la información acerca de varios objetos, identificando las propiedades que comparten los elementos del conjunto
- ☐ Pero no podemos rellenar siempre la información correspondiente a los hechos esenciales, puesto que diferirá para cada objeto concreto
- ☐ Los marcos tienen **ranuras que pueden rellenarse o no**
- ☐ **Las ranuras rellenas representan hechos**

Sistemas de marcos

☐ Representación de conocimiento (base de conocimiento)

- ☐ Conjunto de marcos relacionados mediante los rellenos de las ranuras
- ☐ Marcos **clase** y marcos **instancia** (*ejemplares de las clases*)
 - ☐ Se hace hincapié en la distinción entre individuos y conjuntos
- ☐ Establecimiento de una jerarquía de conceptos con ejemplar/subclase
- ☐ Propiedades y relaciones entre marcos (*representadas mediante las estructuras de ranura/relleno*)

☐ Motor de inferencia

- ☐ Herencia de propiedades, relaciones y procedimientos de cálculo a través de la estructura jerárquica
- ☐ Clasificación de conceptos. Equiparación

Marcos

❑ Tipos de marcos

❑ Marcos clase

- ❑ Representan conceptos, clases, estereotipos, situaciones genéricas
- ❑ Ejemplo: Herramientas, Persona, Coche

❑ Marcos instancia

- ❑ Representan conceptos individuales, objetos, entidades, individuos
- ❑ Ejemplo: Martillo-1, María, M-6595-K

❑ Las propiedades son los atributos de los conceptos y se representan en los marcos como **ranuras** (*slots*)

- ❑ Los valores para estas propiedades son los **rellenos** (*fillers*)

❑ Sistema basado en marcos

❑ Conjunto de marcos clase e instancia unidos por **relaciones**

- ❑ Las relaciones expresan dependencias entre conceptos
- ❑ No hay convenios de estructura fijos

Marcos

❑ Tipos de Relaciones

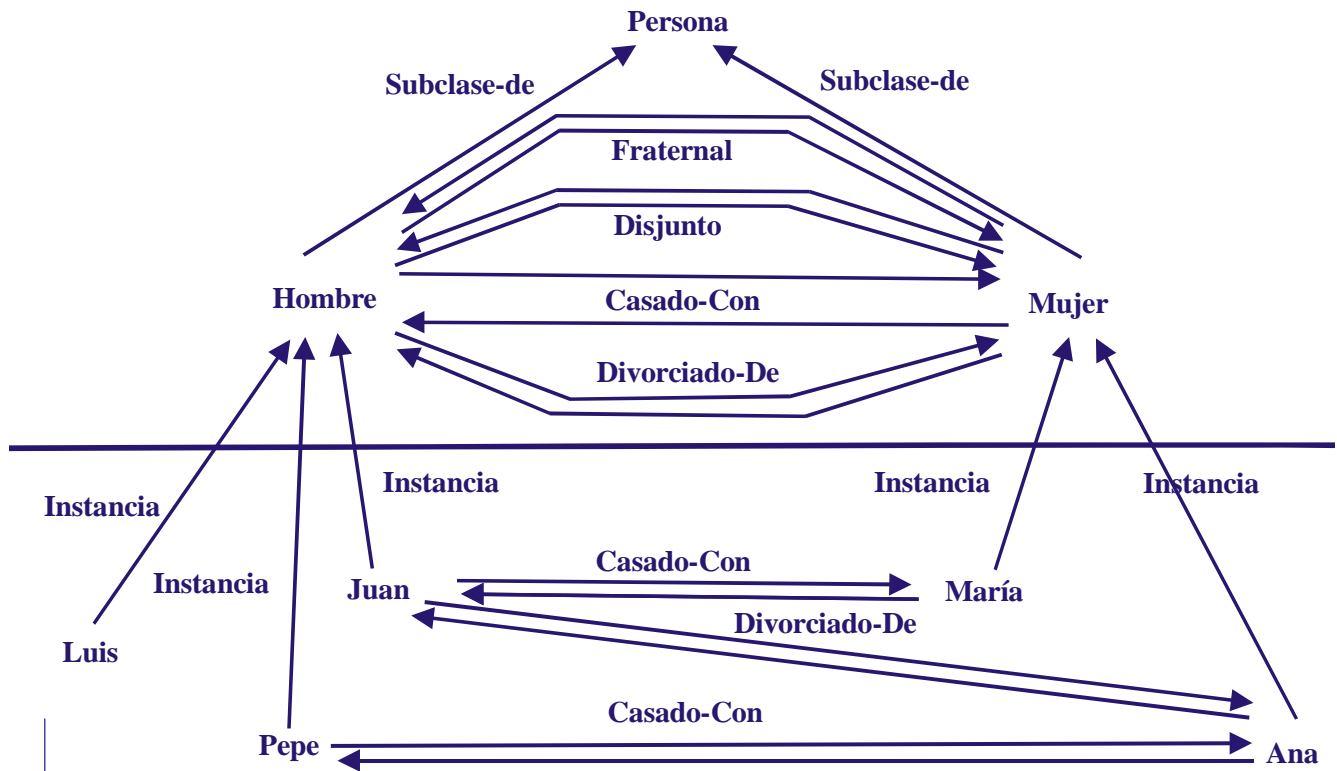
❑ Relaciones estándar

- ❑ *Subclase* y su inversa *Superclase*
- ❑ *Ejemplar* o *Instancia* y su inversa *Contiene*
- ❑ Palabras reservadas (dependientes del sistema)
- ❑ Se suelen manejar inversas de forma automática

❑ Relaciones no estándar

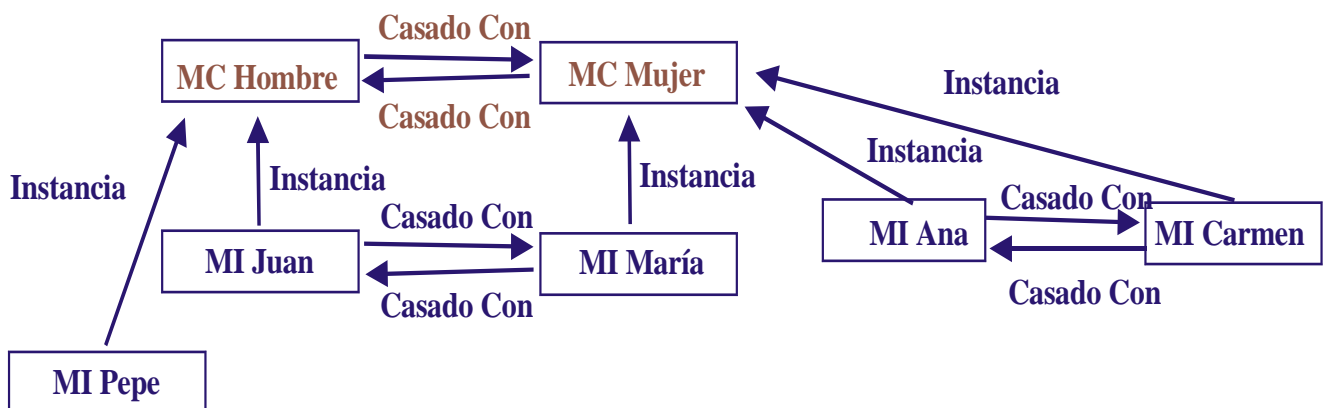
- ❑ Fraternal (hermanos)
- ❑ Disjunto/No Disjunto
- ❑ “a medida” o “ad hoc” (relaciones dependientes del dominio)
- ❑ Las inversas hay que añadirlas

Ejemplo de jerarquía de marcos



Relaciones entre marcos instancia

- ❑ Las relaciones se definen entre marcos clase
- ❑ Los marcos instancia son ejemplares de dichos marcos clase



- ❑ Si el sistema hace comprobación de consistencia podría no aceptarse la conexión mediante la relación *casado_con*

Clases e instancias

MC- ANIMAL

ejemplar: CLASE

...

(*) seDesplaza: SÍ

MI- PIOLIN

ejemplar: AVE

seDesplaza : SÍ

tieneEsqueleto: SÍ

vuela: SÍ

} atributos heredados automáticamente

MC- VERTEBRADO

ejemplar: CLASE

subclase: ANIMAL

(*) tieneEsqueleto: SÍ

MC- AVE

ejemplar: CLASE

subclase: VERTEBRADO

...

(*) vuela: SÍ

- Aunque parece que la herencia sólo funciona a un nivel no es así porque **la relación subclase es transitiva**

$x \in Ave \rightarrow x \in Vertebrado$
 $\rightarrow x \in Animal$

- Cualquier individuo hereda de todas sus superclases (superconceptos)

(*): propiedades heredables por los individuos pertenecientes a la clase

CLASE: palabra reservada que indica que se representa a un conjunto

Propiedades

□ Propiedades de clase

- Son atributos de la clase (o concepto)
- Se *definen y rellenan* en el marco clase
- No son heredables por las instancias
 - Ejemplo: cardinalidad

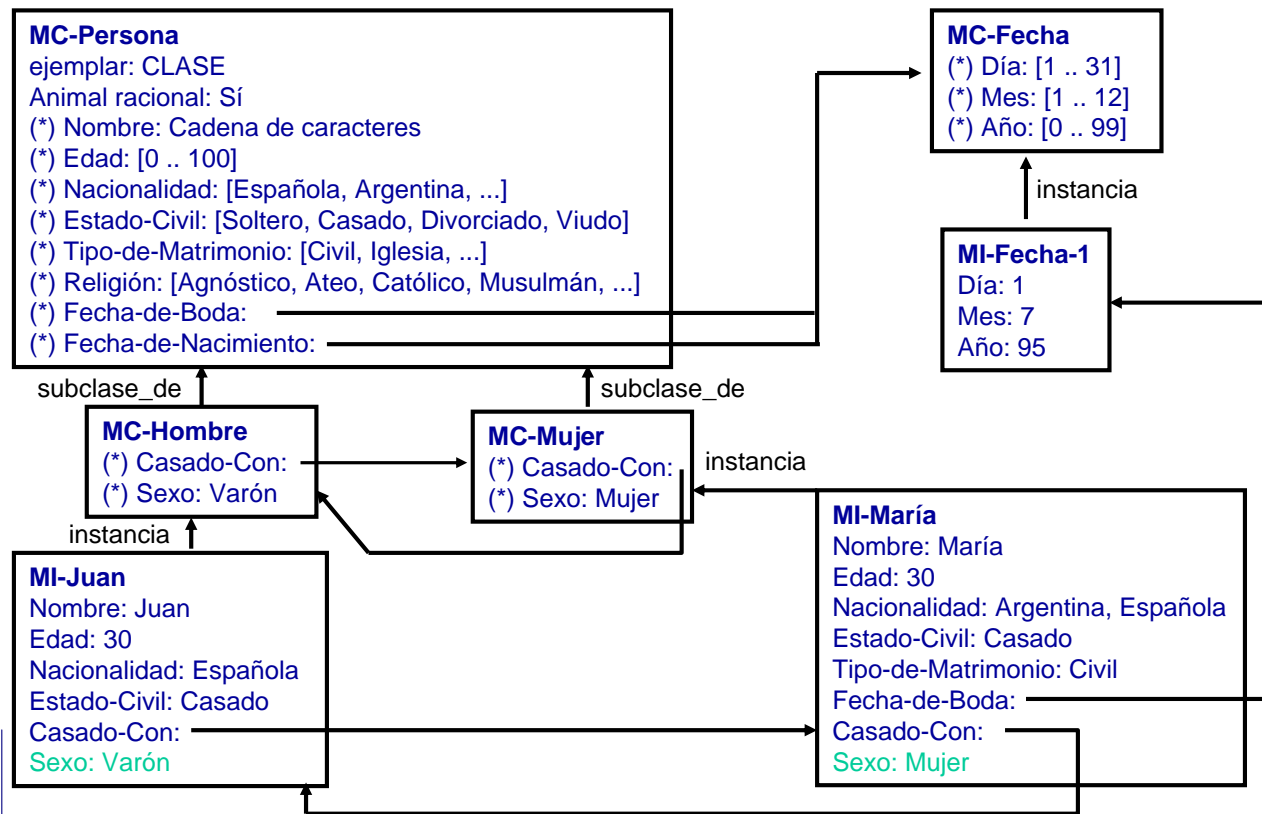
□ Propiedades de instancia

- Son atributos específicos de cada instancia
- Se *definen* en el marco clase
- Si se *rellenan en el marco clase*, todas las instancias heredan su valor (**herencia de valores**)
 - Según las circunstancias, podría ser redefinido en cualquiera de ellas
- Si se *rellenan en el marco instancia*, lo único que se hereda del marco clase es la existencia de la ranura (**herencia de ranuras**)
- Precedidas del símbolo (*)

Formalismo especialmente adecuado para dominios con mucho conocimiento por defecto

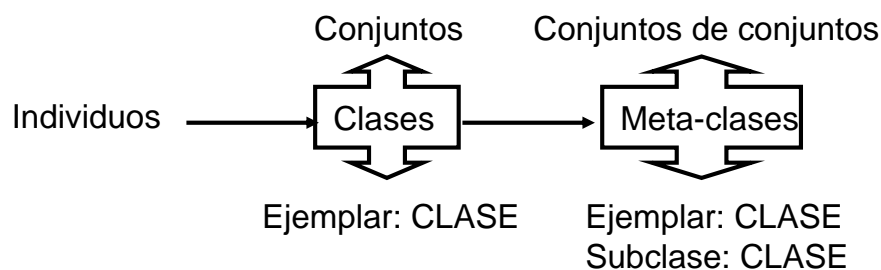


Ejemplo

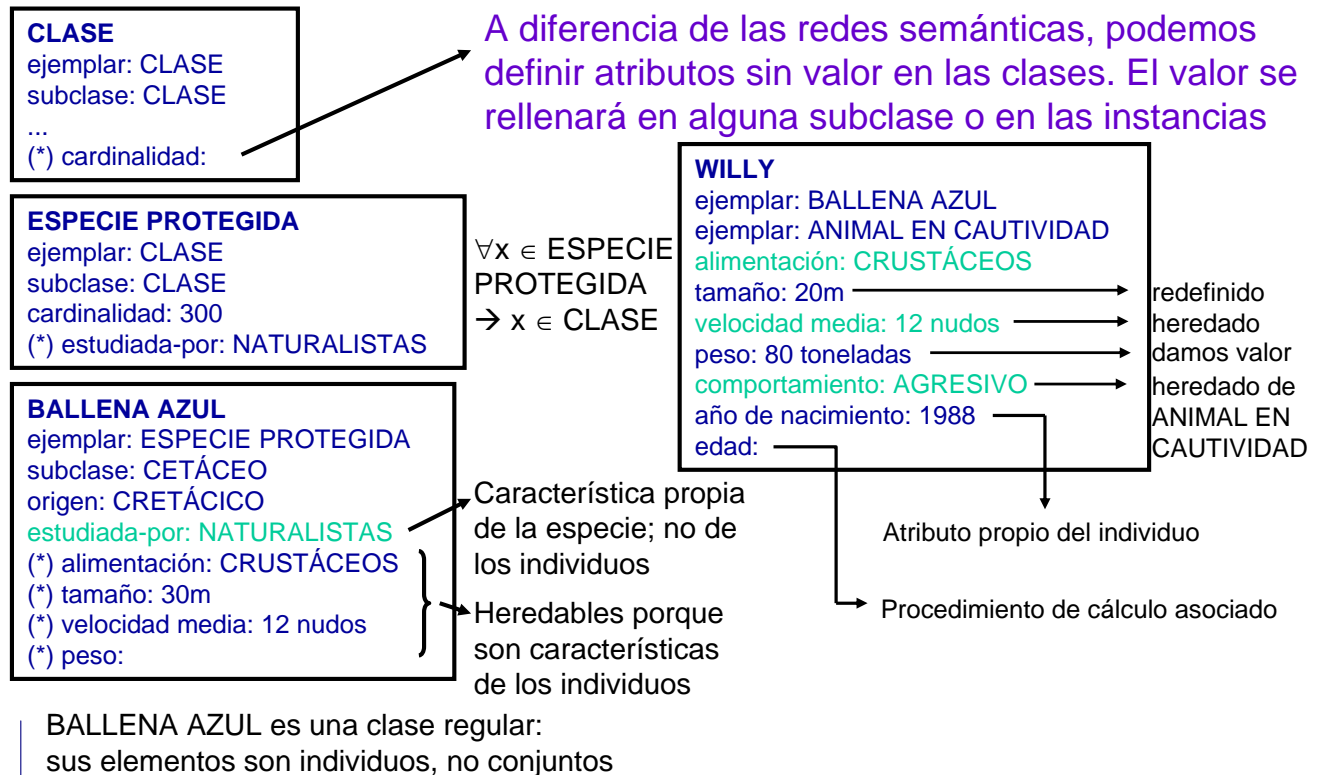


Meta-clases

- ❑ Conjuntos de conjuntos
- ❑ Las instancias de una meta-clase son a su vez clases
- ❑ La manera de caracterizar a las meta-clases es
 - ❑ Son ejemplares de CLASE
(como las clases regulares = conjuntos de individuos)
 - ❑ Son subclases de CLASE
(esto hace que sus instancias sean también clases)



Meta-clases



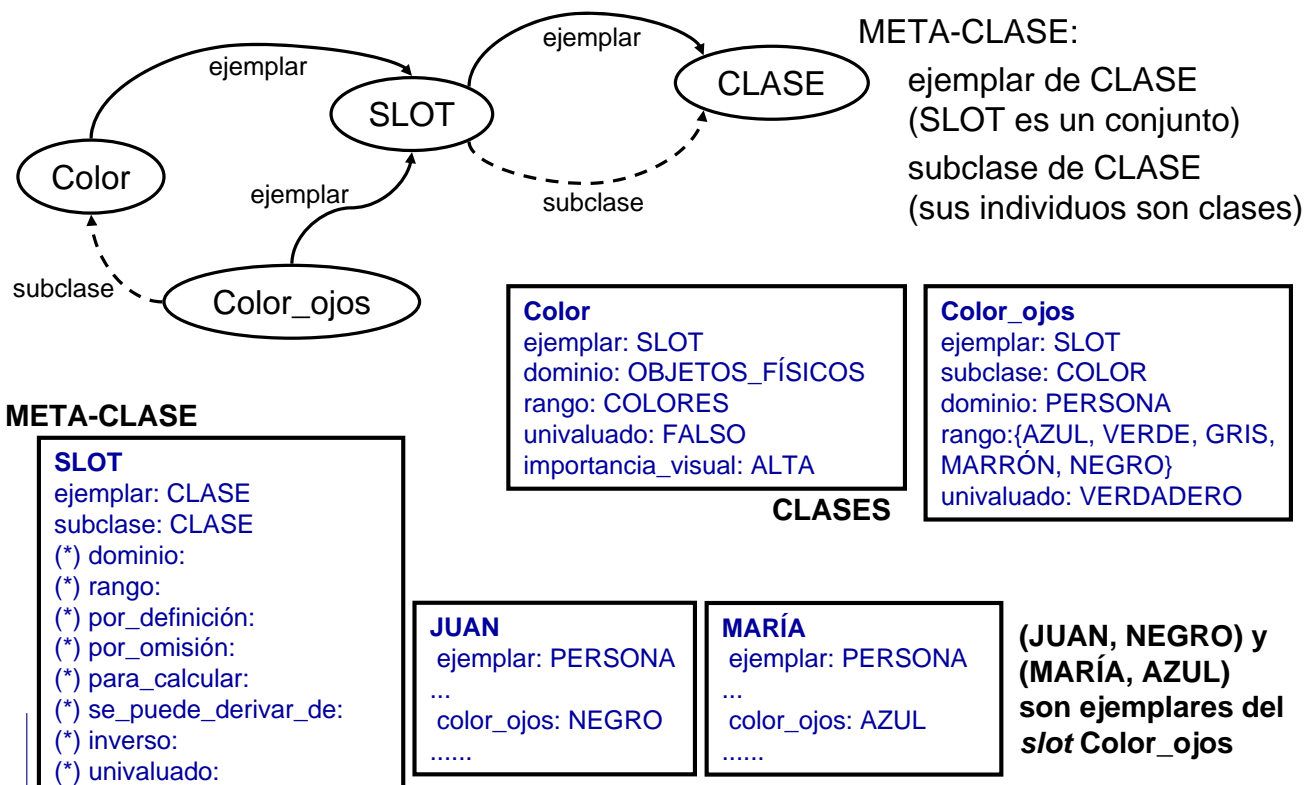
Representación de atributos como marcos

- ☐ Representación del significado o propiedades de los atributos
 - ☐ Lo podemos hacer representando los propios atributos como marcos
- ☐ Cada atributo (ranura) puede ser descrito por una serie de ranuras que se suelen denominar **facetas**:
 - ☐ Dominio
 - ☐ Rango
 - ☐ Valor obligatorio (*por definición*)
 - ☐ Valor por omisión
 - ☐ Reglas de herencia
 - ☐ Reglas o procedimientos para calcular valores de relleno
 - ☐ Inversas
 - ☐ Univaluado/multivaluado
- ☐ Así representamos meta-conocimiento (restricciones sobre el conocimiento a representar en los marcos)

Jerarquías de atributos

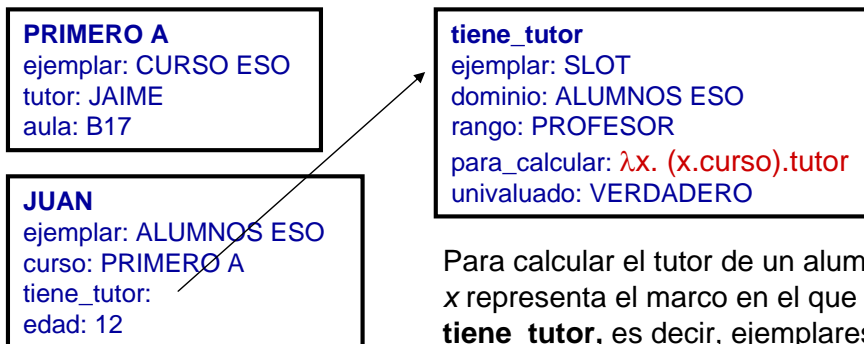
- ❑ Un atributo (ranura o *slot*) es una relación entre los elementos del dominio (las clases para las que tiene sentido) y los elementos de su rango (posibles valores)
 - ❑ Un *slot* es el conjunto de pares ordenados que cumplen esa relación
 - ❑ Atributos como conjuntos de pares: $\{(x, y), (z, u), \dots\}$
- ❑ Un *slot* *S1* puede ser un subconjunto (subclase) de un *slot* *S2*
 - ❑ Por ejemplo, color de ojos \subseteq color
- ❑ La relación de inclusión nos permite crear jerarquías
- ❑ Al conjunto de todos los *slots* lo denominamos SLOT (es una meta-clase)
- ❑ Los sistemas que permiten la representación de *slots* mediante marcos suelen tener restricciones sobre las ranuras definibles (facetas)
- ❑ Las jerarquías de *slots* suelen ser bastante planas

Jerarquías de atributos (*slots*)



Valores calculados

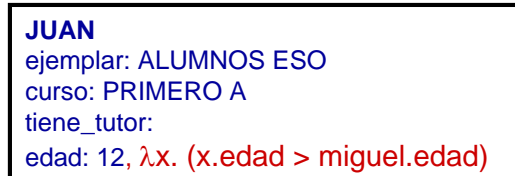
- ❑ Mecanismo general: se representa en el marco del *slot*



Para calcular el tutor de un alumno usamos λ -cálculo: x representa el marco en el que se usa el *slot* **tiene_tutor**, es decir, ejemplares de ALUMNOS ESO
Por ejemplo:

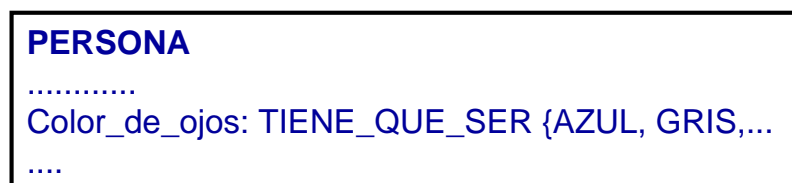
$x \rightarrow x.curso \rightarrow (x.curso).tutor$
JUAN PRIMERO A JAIME

- ❑ Las restricciones particulares de un *slot* para un ejemplar particular se representan en el marco del ejemplar



Representación de atributos sin marcos

- ❑ Hay sistemas que no permiten representar los *slots* como marcos. En ese caso, suelen permitir incorporar algunas restricciones en la definición de los atributos en las clases



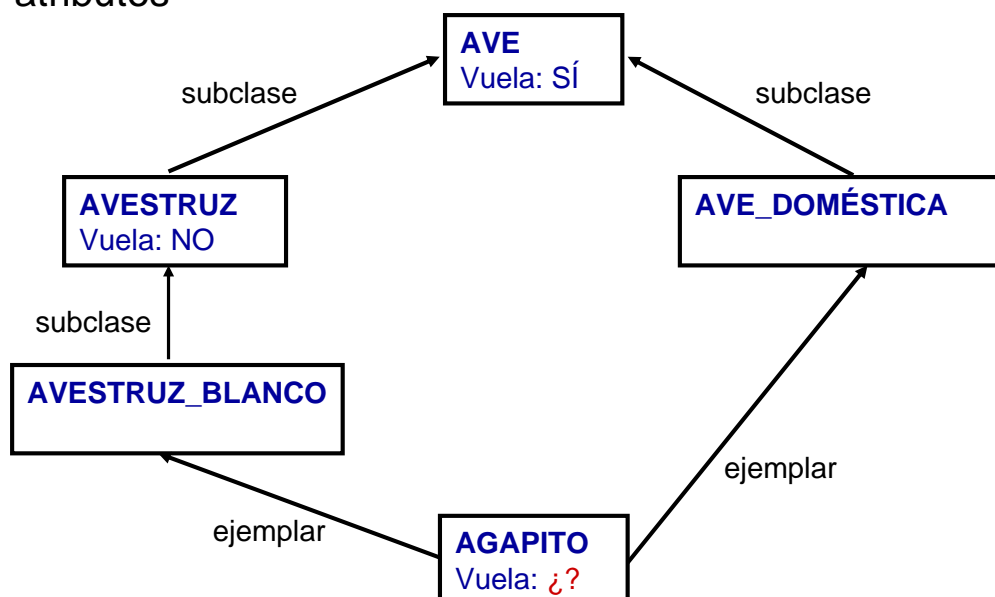
- ❑ TIENE_QUE_SER:
palabra reservada que permite expresar el rango de valores permitidos
- ❑ Otras palabras reservadas
 - ❑ POR_OMISIÓN
 - ❑ POR_DEFINICIÓN
 - ❑ UNIVALUADO

Conocimiento sobre atributos

- ❑ La representación de **meta-conocimiento** sobre los *slots* permite a los sistemas
 - ❑ Realizar control de consistencias en el dominio y el rango de los atributos
 - ❑ Mantener la consistencia entre un atributo y su inverso cuando se cambia uno de ellos
 - ❑ Propagar los valores por definición y por omisión a través de la jerarquía de herencia (*ejemplar* y *subclase*)
 - ❑ Calcular el valor de un atributo cuando se necesita (*para_calcular*, *se_puede_derivar_de*)
 - ❑ Controlar los atributos univaluados

Herencia múltiple

- ❑ Jerarquías: grafos dirigidos acíclicos, en lugar de árboles
- ❑ Distintos antepasados pueden tener distintos valores de los atributos



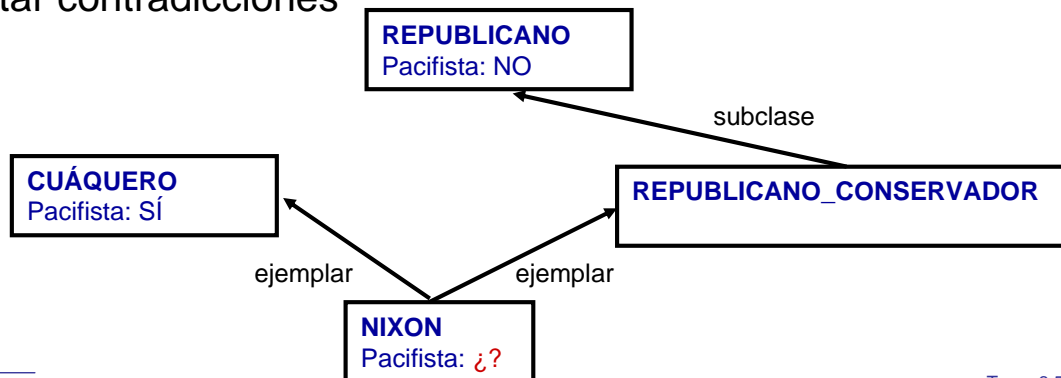
Distancia inferencial [Touretzky, 1986]

❑ Define un orden parcial:

- ❑ Concepto1 está más cerca de Concepto2 que de Concepto3 **si y sólo si** Concepto1 tiene un camino de inferencia a través de Concepto2 hasta Concepto3 (es decir, Concepto2 está entre Concepto1 y Concepto3)

$$\text{distancia}(\text{Concepto1}, \text{Concepto2}) < \text{distancia}(\text{Concepto1}, \text{Concepto3}) \\ \Leftrightarrow \exists \text{camino}(\text{Concepto1}, \text{Concepto2}, \text{Concepto3})$$

- ❑ La distancia inferencial no siempre es aplicable → permitirá detectar contradicciones



Herencia de propiedades: algoritmo

- ❑ Para obtener el valor desconocido V de un atributo A en una instancia I

- ❑ $CANDIDATOS := \emptyset$

- ❑ Búsqueda 1º en profundidad en la jerarquía a partir de I de todos los superconceptos SC (en orden ascendente)

- ❑ Si en SC se encuentra un valor para A se añade a $CANDIDATOS$ y se finaliza con esa rama
 - ❑ Si en SC no se encuentra ningún valor, ascendemos otro nivel. Si no hay más niveles, terminamos con esa rama

- ❑ Para cada elemento C de $CANDIDATOS$:

- ❑ Si existe algún otro elemento de $CANDIDATOS$ que ha sido obtenido de un concepto que esté a menor distancia inferencial de I que el concepto del que se ha obtenido C , entonces sacar C del conjunto de $CANDIDATOS$

- ❑ Si el cardinal de $CANDIDATOS$ es:

- ❑ 0: no se ha obtenido ningún valor
 - ❑ 1: se devuelve el único elemento de $CANDIDATOS$ como V
 - ❑ >1 y todos sus elementos son iguales: devolver el valor como V
 - ❑ >1 y elementos distintos: informar de que hay una contradicción

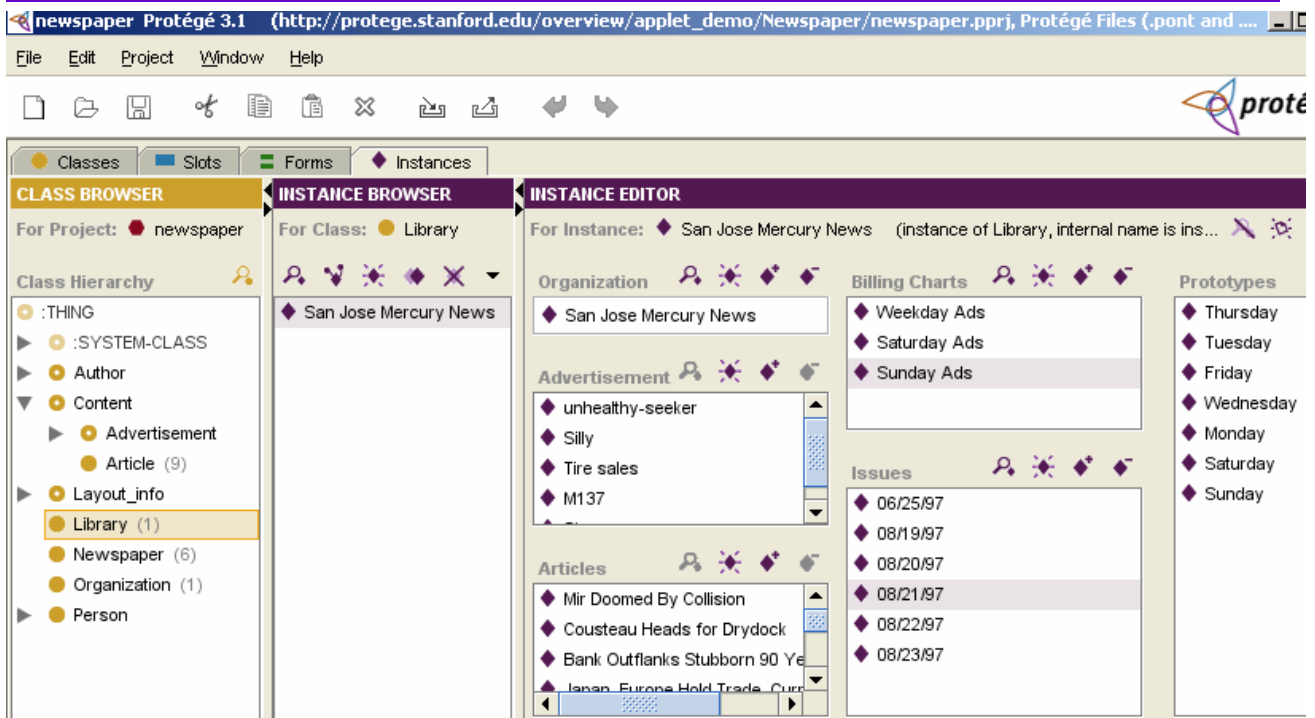
Representación de marcos en Prolog: convenios

- ☐ Ranuras definidas explícitamente en los marcos clase
`slot(NombreMarco, NombreRanura).`
- ☐ Rellenos de las ranuras
`valor(NombreMarco, NombreRanura, Valor).`
- ☐ Gestión de la herencia
 - ☐ Predicados que la implementen teniendo en cuenta los distintos tipos de herencia
 - ☐ Herencia habitual: *es_un, instancia o ejemplar, subclase*
`ejemplar(NombreMarcoInstancia, NombreMarcoClase).`
`subclase(NombreMarcoClase, NombreMarcoClase).`
 - ☐ Herencia de ranuras
 - ☐ Herencia de valores

Ventajas de los marcos

- ☐ Facilitan el **razonamiento basado en expectativas**
 - ☐ Un *slot* es un lugar donde se espera un cierto tipo de valor dentro del contexto de un marco
 - ☐ Proporcionando un lugar para el conocimiento, se crea la **posibilidad del conocimiento incompleto o inexistente**, permitiendo el razonamiento basado en intentar confirmar expectativas
 - ☐ Se ha aplicado en sistemas de comprensión del lenguaje natural
- ☐ Posibilidad de **asociar procedimientos de cálculo a los atributos**
 - ☐ Mecanismo hacia atrás que permite rellenar atributos “cuando se necesita” (el procedimiento “enganchado” al *slot* se dispara al preguntar por su valor)
 - ☐ Mecanismo hacia delante para rellenar atributos “cuando se añade” (cuando se rellena un *slot*, todos los *slots* de otros marcos que dependan de él se rellenan automáticamente)
- ☐ **Representación estructurada del conocimiento**, incluso en el caso del conocimiento procedimental
 - ☐ La fase de equiparación o *matching* para determinar qué procedimiento o regla aplicar se realiza aquí mediante un proceso de clasificación

Software de Marcos : Protege-Frames



<http://protege.stanford.edu/download/registered.html> (versión 3 “basic” incluye “frames”)

demo: <http://protege.stanford.edu/overview/protege-frames.html>

IAIC – Curso 2008-09

Tema 3.5 - 29

Ejemplo de Marco en Control de Tráfico

```
(DEFINE-FRAME M30-CONGESTION-AT-ODONNELL
:TYPE MADRID-PROBLEM-FRAME
:KB M30-FROM-NII-TO-NIII-PROBLEM-FRAMES
:MAX-CM 1
:CONFIRMING
)
;; STATE OF DETECTORS
((SPEED-> E17L INTO 50 AND 180) :IV 1)
((OCCUPANCY-> E17L INTO 0 AND 40) :IV 1)
((SPEED-> E17C INTO 50 AND 180) :IV 1)
((OCCUPANCY-> E17C INTO 0 AND 40) :IV 1)
((SPEED-> E19L INTO 0 AND 50) :IV 1)
((OCCUPANCY-> E19L INTO 40 AND 100) :IV 1)
((TGRAD-SPEED-> E19L INTO -180 AND -20) :IV 1)
((TGRAD-OCCUPANCY-> E19L INTO 20 AND 100) :IV 1)
((SPEED-> E19C INTO 50 AND 180) :IV 1)
((OCCUPANCY-> E19C INTO 0 AND 40) :IV 1)
((SPEED-> E21L INTO 50 AND 180) :IV 1)
((OCCUPANCY-> E21L INTO 0 AND 40) :IV 1)
((SPEED-> E21C INTO 50 AND 180) :IV 1)
((OCCUPANCY-> E21C INTO 0 AND 40) :IV 1)
;; STATE OF DETECTION AREAS
((SGRAD-SPEED-> M30-AT-ODONNELL INTO 40 AND 180) :IV 1)
((SGRAD-OCCUPANCY-> M30-AT-ODONNELL INTO -100 AND -30) :IV 1)
;; STATE OF PROBLEM FOCUSES
:FOCUSES
'((M30-AT-NII
:STATUS FREE)
(M30-AT-BRASILIA
:STATUS FREE)
(M30-BETWEEN-NII-AND-ALCALA
:STATUS FREE)
(M30-AT-ALCALA
:STATUS FREE)
(M30-AT-ODONNELL
:STATUS UNEXPECTED-EVENT
:EXCESS (1000 1300)
:PARTICIPATION
(((10 20) M30-NORTH->ODONNELL->
M30CENTRAL-AT-ALCALA)
((30 60) M30-NORTH->M30-AT-NIII->
M30SIDE-AT-ODONNELL)
((0 10) NII->ODONNELL
M30CENTRAL-AT-BRASILIA->
M30CENTRAL-AT-ALCALA)
((10 30) NII->M30-AT-NIII->
M30CENTRAL-AT-BRASILIA->
M30SIDE-AT-ODONNELL)
((0 10) ALCALA->ODONNELL)
((10 20) ALCALA->M30-AT-NIII->
M30SIDE-AT-ODONNELL)))
(M30-BETWEEN-ALCALA-AND-ODONNELL
:STATUS FREE)))
```

IAIC – Curso 2008-09

Tema 3.5 - 30