Métodos de Inteligencia Artificial

L. Enrique Sucar (INAOE)

esucar@inaoep.mx

ccc.inaoep.mx/esucar

Tecnologías de Información

UPAEP

Representaciones estructuradas

- Introducción
- Redes semánticas
- Prototipos Marcos (*frames*)

Introducción

Las limitaciones de las representaciones en base a reglas, en particular, la necesidad de representar aspectos como estructura y relaciones, llevaron a otros esquemas que en general englobamos como representaciones estructuradas.

Introducción

Dentro de este tipo de representaciones las dos más significativas, son:

- **Redes Semánticas**
- **Prototipos o Marcos (*frames*)**

Introducción

Estas representaciones se basan en el uso de grafos, es decir, en representaciones en base a nodos y sus relaciones.

Representación con grafos

- Representación de relaciones entre conceptos (redes semánticas o asociativas).
- Representación de jerarquías para discriminación y clasificación (sistemas de *frames*).

Redes Semánticas

Modelo de memoria humana para capturar la semántica de las palabras y lograr un uso del significado parecido a los humanos [Quillian'66].

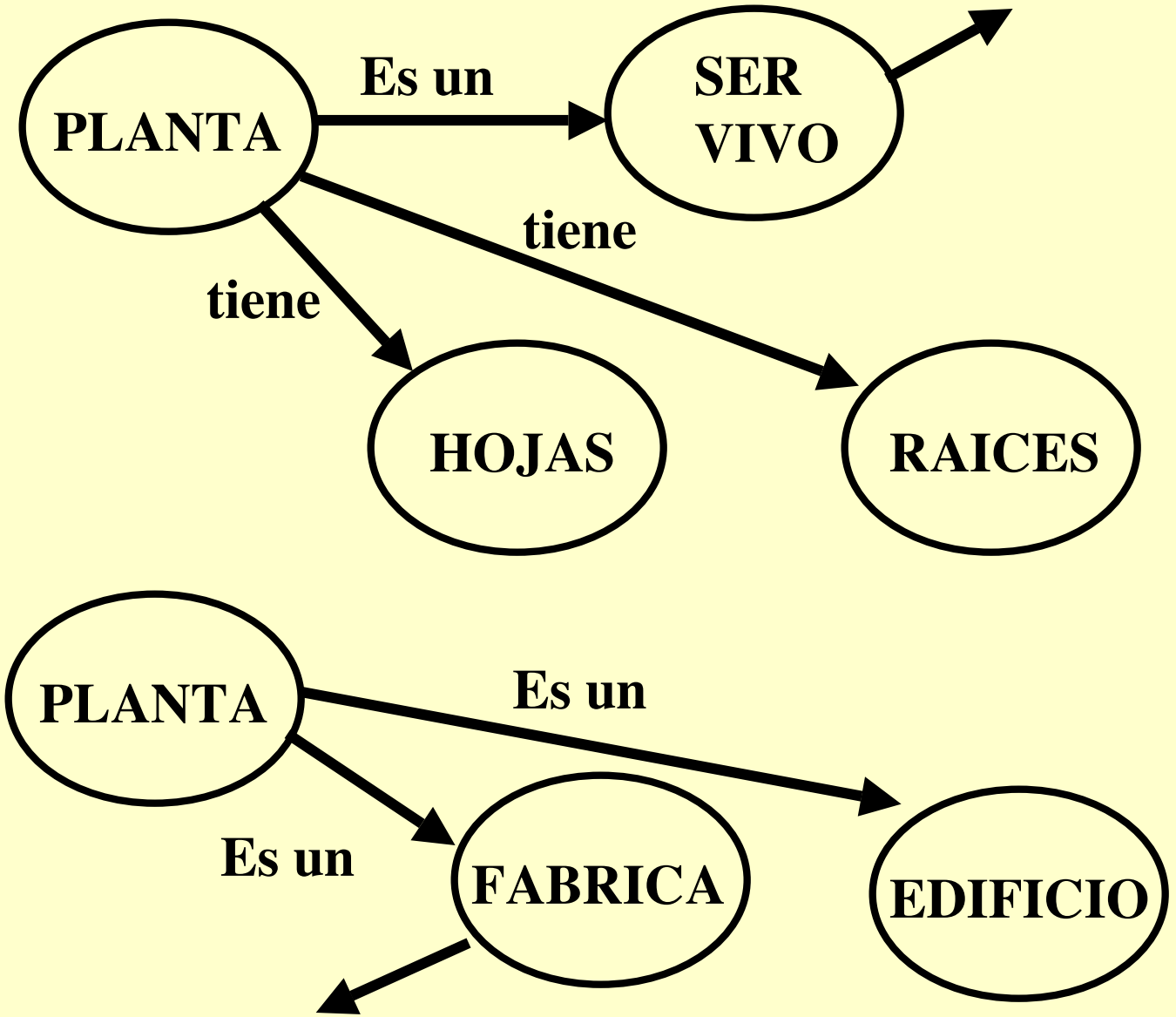
Un tipo de red en la cual los nodos representan objetos, conceptos o situaciones y los arcos representan relaciones entre ellos.

Se llama red semántica porque se usaron originalmente para representar el sentido en expresiones de lenguaje natural.

Ejemplo: Definición en un Diccionario

PLANTA
(1) cosa viviente con hojas y raíces
(2) fábrica o edificio industrial

FÁBRICA
lugar donde se fabrican bienes con máquinas


PLANTA
Es un
SER VIVO
tiene
tiene
HOJAS
RAICES
PLANTA
Es un
Es un
FABRICA
EDIFICIO

Representación

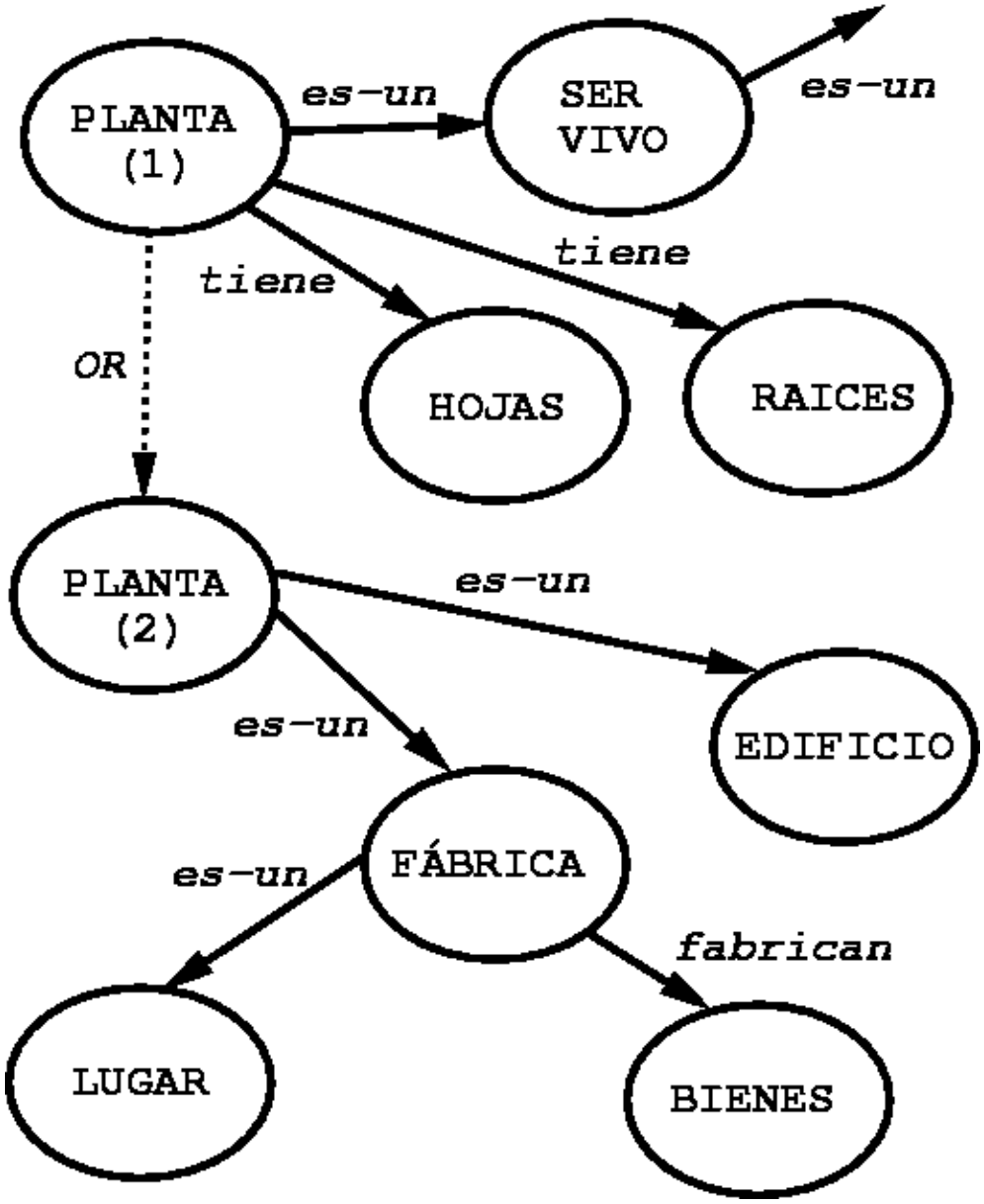
- Los nodos: conceptos de palabras
- Los arcos: ligan conceptos para establecer la definición

- Cada palabra o nodo conceptual se consideraba la cabeza de un "plano" que tiene su definición (p. ej., si *banco* tiene 3 significados, entonces existen 3 planos para él).

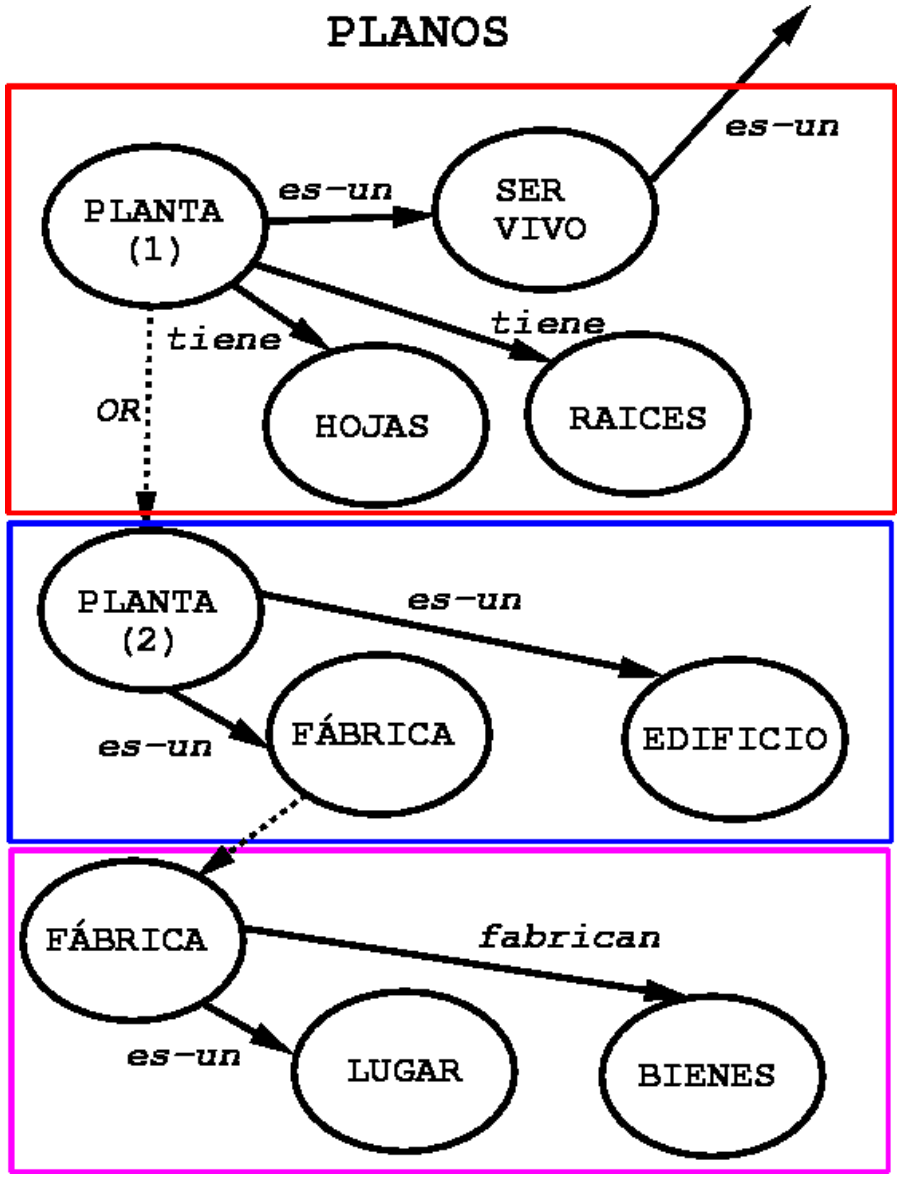
Las ligas en el plano representan su definición.

Apuntadores fuera del plano hacen referencia a otros objetos (y planos) en donde se definen.

RED SEMÁNTICA



Ejemplo de Diferentes Planos



Relaciones

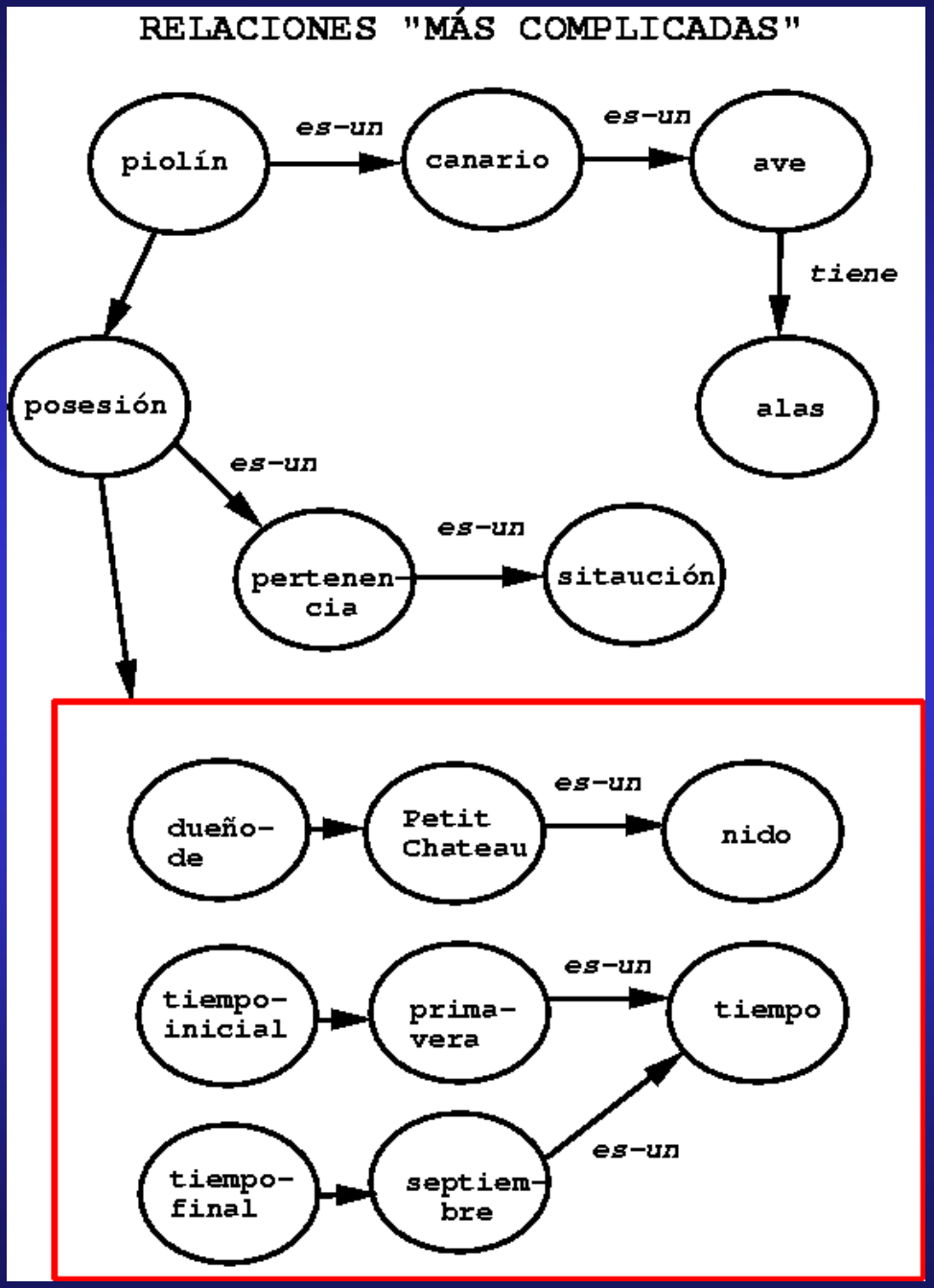
Existen 2 tipos de ligas principales:

- subclase (*is-a*): las clases de "arriba" están definidas en términos de conceptos generales que se asumen que se cumplen en todas sus subclases
- modificadores: propiedades particulares de conceptos específicos.

Pueden existir apuntadores a: superclases (*is-a*), modificaciones, disjunciones, conjunciones y sujeto/objeto.

Puede existir herencia (v.g., un canario es un animal) y herencia de propiedades (v.g., un canario come).

Case Frame


RELACIONES "MÁS COMPLICADAS"
piolín
es-un
canario
es-un
ave
tiene
alas
posesión
es-un
pertenen-cia
es-un
sitaución
dueño-de
Petit Chateau
es-un
nido
tiempo-inicial
prima-vera
es-un
tiempo
tiempo-final
septiem-bre
es-un

Case Frames

El permitir tener un conjunto de arcos de salida se llama “case frame”, el cual agrupa información de un concepto

posesión es una instancia de *pertenencia* y hereda los arcos del “case frame”.

Las redes semánticas permiten tener valores por *default* y cierta expectación acerca de los posibles valores de un atributo.

Inferencia

La idea de ésta representación fue originalmente la interpretación del lenguaje.

Quillian describe dos formas de uso: encontrar las similitudes y diferencias entre palabras, y expresar oraciones congruentes en base a la información en la red.

En general podemos usar este tipo de estructuras para diferentes tipos de razonamiento.

Tipos de Razonamiento

1. Búsqueda asociativa:

Encontrar si están relacionados dos o más conceptos, y su tipo de relación mediante el seguimiento de la red hasta encontrar las interacciones.

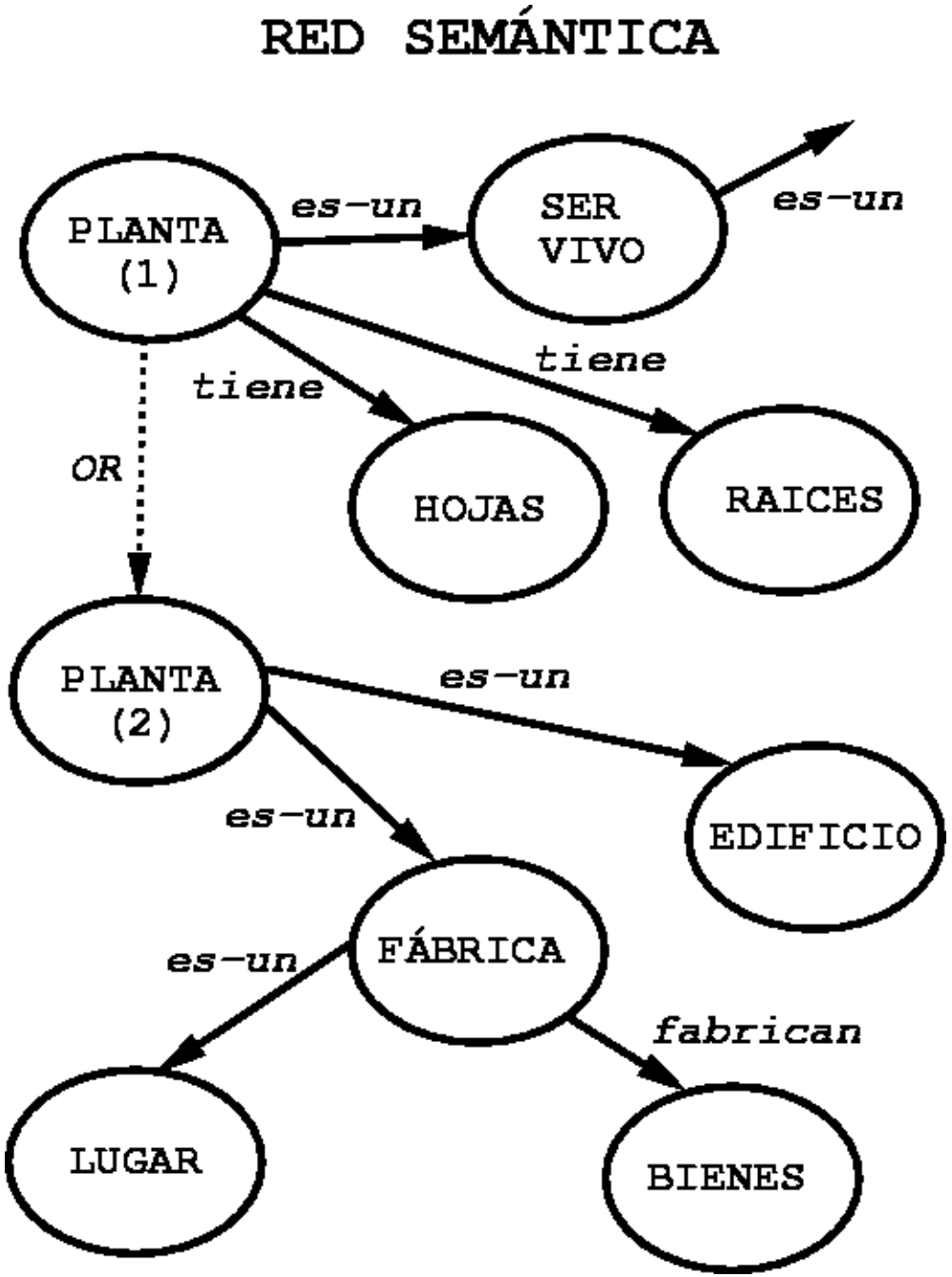
2. Reconocimiento:

Dada una serie de características (nodos “token”) encontrar el concepto (nodo clase) que mejor las define mediante su búsqueda y seguimiento en la red (reconocimiento de imágenes).

3. Descripción:

Expresar un concepto en base a sus componentes y relaciones entre ellas (lenguaje natural).

Ejemplos de Inferencia

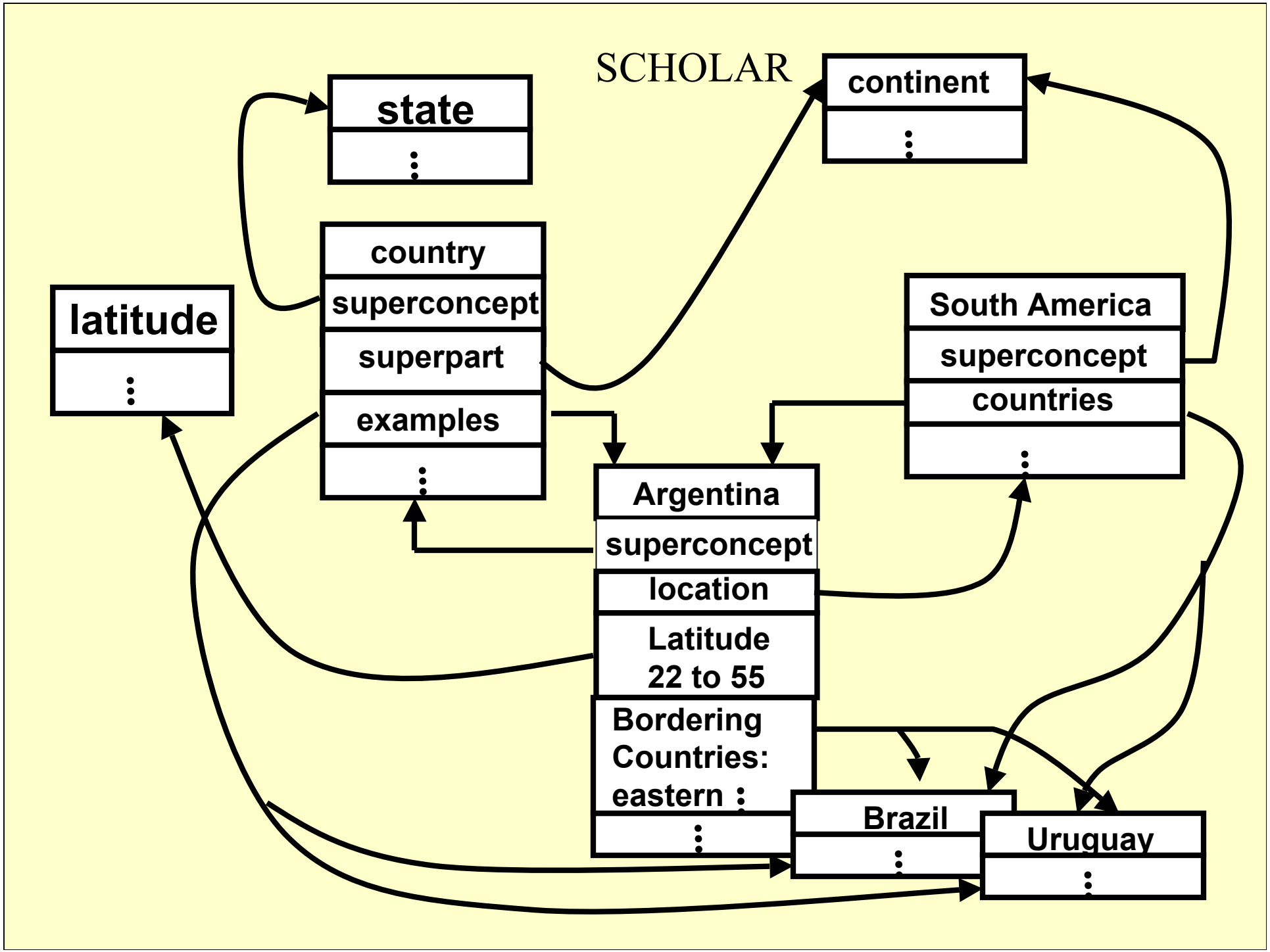


Ejemplos de sistemas basados en redes semánticas.

SCHOLAR (Carbonell): Uno de los primeros sistemas que usaron redes semánticas. Se aplicó para enseñar la geografía de Sudamérica.

Carbonell distingue entre:

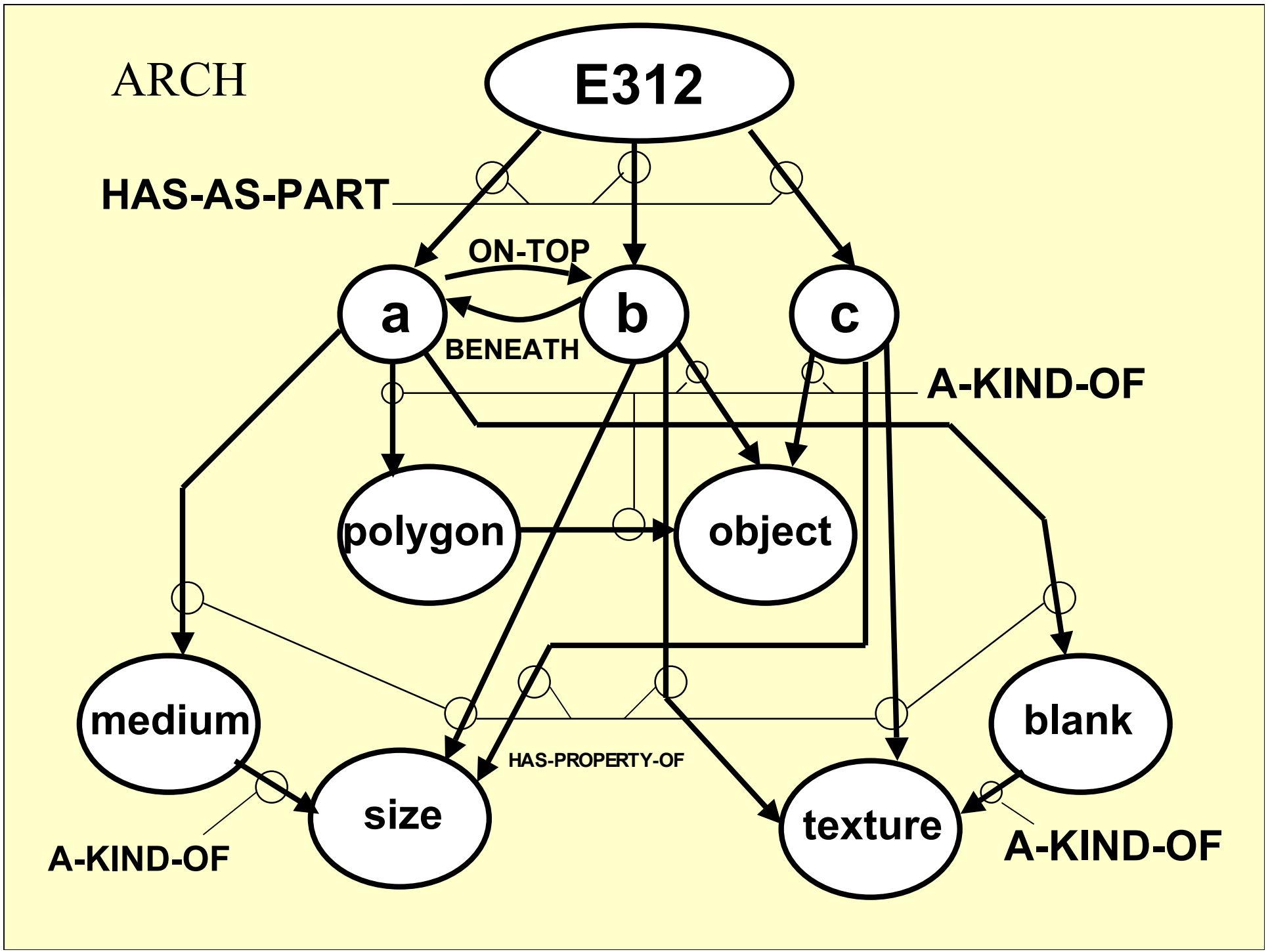
unidades conceptuales (clases) y unidades de ejemplos (instancias).


SCHOLAR
state
continent
country
superconcept
superpart
examples
latitude
South America
superconcept
countries
Argentina
superconcept
location
Latitude
22 to 55
Bordering
Countries:
eastern
Brazil
Uruguay

Otro Ejemplo

ARCH (Winston): sistema para aprender conceptos de estructuras físicas a partir de ejemplos de estructuras descritos en forma de redes semánticas.

El proceso de generalización permite cambiar relaciones entre objetos.


ARCH
E312
HAS-AS-PART
ON-TOP
a
b
c
BENEATH
A-KIND-OF
polygon
object
medium
blank
HAS-PROPERTY-OF
size
texture
A-KIND-OF
A-KIND-OF

Redes Semánticas en el Web: *Semantic Web*

- Incoporar "significado" a la información en el WWW:
  - Ontologías de conceptos en diversos dominios
  - Relaciones entre conceptos
- Esto facilitará a agentes el "entender" la información y hacer búsquedas mucho más sofisticadas
- Uso de estándares como XML y RDF

Ventajas

- Representación "estructurada" del conocimiento.
- *Economía cognoscitiva*: no es necesario representar en forma explícita todas las propiedades.
- Definición de *distancia semántica* entre conceptos (número de ligas a recorrer).
- Representación "analógica" de conocimiento.

Problemas

- Las redes semánticas no son muy escrupulosas en cuanto al significado de los nodos (v.g., perro se refiere a la clase, el concepto o un perro en particular).

Problemas

- Para establecer si existe relación entre dos conceptos, se sigue un proceso de búsqueda de intersección. Esto, sin embargo, no evita la explosión combinatoria.

Problemas

- Finalmente una red semántica tiene: nodos, arcos y reglas de combinación (sintáxis) y lo que significan (semántica).

- El problema es por falta de distinción entre lo *intensional* (sense/meaning) y *extensional* (reference/denotation), por ejemplo:

rojo: todas las cosas rojas (extensional)
la propiedad de ser rojo (intensional)

Implementación

Representación alternativa a lógica en forma de cláusulas restringidas a predicados binarios.

Predicados binarios representan las relaciones correspondientes a los arcos en el grafo: el símbolo del predicado corresponde a la etiqueta del arco, los argumentos del predicado corresponden a los vértices incidentes de dicho arco.

Ejemplos:

isa(tanque, componente)

pared(tanque, acero)

Implementación

Esta representación se puede extender utilizando variables y predicados no aterrizados. De esta forma una cláusula representa una “subred”.

Ejemplos:

pared(X, acero) ← isa(X, tanque)

contenido(X, agua) ← isa(X, tanque)

isa(X, tanque) ← isa(X, tanque-domo)

En principio no es un problema la restricción a predicados binarios, ya que un predicado de grado $n > 2$ se puede representar como $n + 1$ predicados binarios.

FRAMES

Frames: estructuras de datos representando situaciones prototípicas (Minsky '75)

Una de las ideas intuitivas detrás de los Frames, es que la memoria se basa mucho en estereotipos (propiedades típicas de los objetos)

Los sistemas de *frames* razonan acerca de clases de objetos usando representaciones prototípicas, pero que pueden modificarse para capturar las complejidades del mundo real.

Representación

Idea: tener una sola estructura de datos para poner el conocimiento relevante acerca de una clase de objetos, en lugar de tener el conocimiento distribuido en forma de reglas o fórmulas lógicas.

Permite construir conocimiento declarativo y procedural en un registro con *slots* y *fillers* o *facets*

Los *slots* son atributos y los *fillers* o *facets* son los valores, v.g.,

(frame (nombre camión)
(is-a objeto)
(color rojo)
(llantas 10)
...)

Los slots pueden tener valores múltiples

Jerarquías de *Frames*

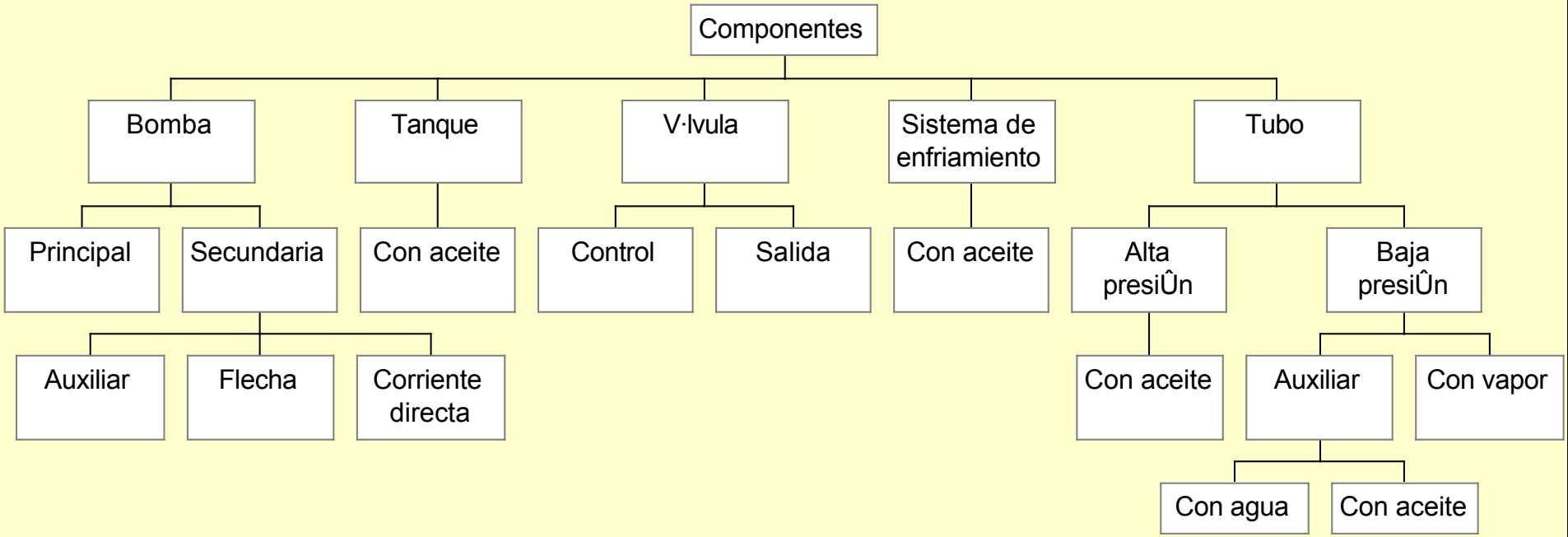
Frames están puestos en una jerarquía en donde los frames de "abajo" pueden heredar los valores de los *slots* de los frames de "arriba"

Normalmente la herencia se hace por medio de los arcos: *is-a* (al final *instance-of*)

En general los frames de “arriba” tienen información típica (poco variable) mientras que los de “abajo” tienen información más específica. En ausencia de ésta, se utiliza la de los padres.

Se pueden hacer deducciones a través de la jerarquía (se distinguen entre los frames clases o genéricos y los frames instancias).

Ejemplo de Jerarquía



Jerarquía de los componentes de los sistemas de lubricación de una planta de energía.

Excepciones, *Defaults* y *Demons* (facets/fillers)

El permitir que un *slot* esté presente en más de un frame nos permite manejar excepciones (v.g., pingüino).

Se puede tener información adicional, como: procedimientos para calcular el valor de un slot cuando no se tiene, procedimientos para actualizar valores de un slot cuando un valor de otro slot es actualizado, restricciones en los valores que puede tener un slot, etc.

Procedimientos

Datos, definiciones y procedimientos están agrupados en módulos que pueden compartir información y procedimientos por medio de mecanismos de herencia.

Los fillers o facets pueden tener varias formas de calcular un valor: *value, default* y *demons*.

Pegados a los *slots* pueden existir procedimientos que se activan cuando el slot es accesado o actualizado.

Ejemplo - VALUE/DEFAULT

VALUE: (color (valor rojo))

DEFAULT: si no tiene un valor, toma el de default, e.g.,

```
(frame coche
        (color (valor ?))
        (llantas (valor ?)
                 (default 4)))
```

Ejemplo - DEMONS/MÉTODOS

IF-NEEDED: si no tiene un valor y se necesita, se invoca al procedimiento escrito en el *facet if-needed* (éste podría ser preguntarle al usuario, por ejemplo)

```
(frame tanque
                (largo (valor 3))
                (ancho (valor 5))
                (area (valor ?)
                        (if-needed (func-area (ancho largo)))))


(frame tanque1
                    (is-a tanque)
                    (largo (valor 2))
                    (ancho (valor ?))
                    (area (valor ?)))

(defun func-area (A L)
                     (* A L))
```

Ejemplo - DEMONS/MÉTODOS

IF-ADDED: al añadir un valor en un slot
se puede activar un procedimiento
(el cual puede afectar el valor de otro slot)

```
(frame tanque1
        (área (valor ?)
                (if-added (mult*2 área)
                        (imprime: el doble del
    área))))
```

IF-REMOVED: al quitar un valor de un slot se activa un procedimiento.

También se puede tener: *before* y *after*, los cuales se activan antes y después de obtener un valor.

Inferencia

1. Reconocimiento: dados ciertos valores (atributos) encontrar el Frame.
2. Valores típicos/Demons: deducir información faltante de un Frame
3. Herencia: obtener información de instancias o subclases a partir de sus ascendientes.

Estrategias

1. Valores, *defaults*, *demons* en un nivel, y luego hacia arriba (herencia-Z).

Idea: los valores que se puedan obtener en un nivel son más confiables que los de sus niveles superiores.

2. Valores hacia arriba, *defaults* hacia arriba y *demons* hacia arriba (herencia-N).

Idea: si se puede obtener un valor es más confiable del que se obtenga por *default* o por medio de los *demons*.

Procedimiento de Herencia

Sea F un frame y S un slot
UNTIL se encontró un valor para S o F = nil
IF F tiene un valor para S acaba
ELSE sea F = superclase de F por medio del slot IS-A

Con default o demon sería:

Sea F un frame y S un slot
UNTIL se encontró un valor para S o F = nil
IF F tiene un (demon/default) para S
Then (ejecuta el demon/asigna el default) y acaba
ELSE sea F = superclase de F por medio del slot IS-A

Herencia-Z:

Para combinar: valor, demon, default

Sea F un frame y S un slot
UNTIL se encontró un valor para S o F = nil
IF F tiene un valor para S Then asigna el valor
ELSE IF F tiene un demon, Then ejecuta el demon.
ELSE IF F tiene un default para S, Then usa el default
ELSE sea F = superclase de F por medio del slot IS-A

Herencia-N:

Realiza:

- **herencia con valor**
- **herencia con demons**
- **herencia con defaults**

Herencia Múltiple y Ambigüedad

A veces se quiere heredar información de más de un frame (la organización se vuelve más una red que un árbol).

Con herencias múltiples no se tiene problemas mientras no exista conflicto en la información.

En herencia múltiple se tiene que incluir un método que decida de donde heredar.

Manejo de Conflictos

Algunos sistemas no deciden (escépticos) por ser contradictorios.

Algunos permiten varias conclusiones (crédulos)

Se puede usar información adicional para resolver la ambigüedad (v.g., con un demonio: IF-NEEDED)

En algunos sistemas se permiten que los valores de slots apunten a otros frames

A veces se cancelan líneas de herencia para eliminar ambigüedades (*preclusión*).

Ejemplo de Herencia Múltiple

Análisis

Los Frames tienen ventajas y desventajas similares a las redes semánticas.

Redes Semánticas – Frames

- Se pueden ver los frames como una forma modular de red semántica, con ligas a atributos (dentro del frame) y a super-clases (fuera del frame). Ejemplo:

Implementación

Podemos representar un “frame” en lógica como una serie de predicados aterrizados (hechos). Existen varias formas de hacerlo, tres posibles son:

1. frame(objeto,atributo,valor)
2. objeto(atributo,valor)
3. atributo(objeto,valor)

Tarea

- Leer Capítulo12 (12.5-7) de Russell