Representación del conocimiento en Sistemas de Marcos

Herencia simple sin excepciones

Herencia simple con excepciones

Facets

Contenido

- Origen
- Modelo básico
- Herencia simple sin excepciones
- Relación Marcos/ lógica primer orden
- Herencia simple con excepciones
- Extensión de la representación: facets
- Ejemplo representación: sumador completo
- Integración de LRC: marcos y reglas

Origen

- Minsky, 75, “A Framework for representing knowledge”

“... Esta es la esencia de la teoría de los marcos: cuando nos enfrentamos a una nueva situación, o cuando realizamos un cambio sustancial en nuestra visión de un problema, seleccionamos en nuestra memoria una estructura denominada marco. Este es un esquema que nosotros recordamos y que debemos adaptar a la realidad cambiando los detalles según sea necesario...”

- Estructuras de datos utilizadas para representar elementos bien conocidos (prototípicos)
- Para adaptarnos a la situación actual, accedemos a la estructura que más se asemeja, y modificamos los detalles necesarios

Modelo básico sistema de marcos

- Marcos: clase y particularización
- Relaciones: clase-superior y particularización-de
- Se impone estructura de grafo dirigido acíclico

suficiente para establecer Jerarquía de Marcos (de herencia, Taxonomía ,...)

- Restricción adicional:estructura de árbol dirigido (herencia simple)

Tipos de marcos y relaciones jerárquicas

- Marcos de Clase (MC)

  conocimiento relativo a clases
- Marcos de Particularización (MP)

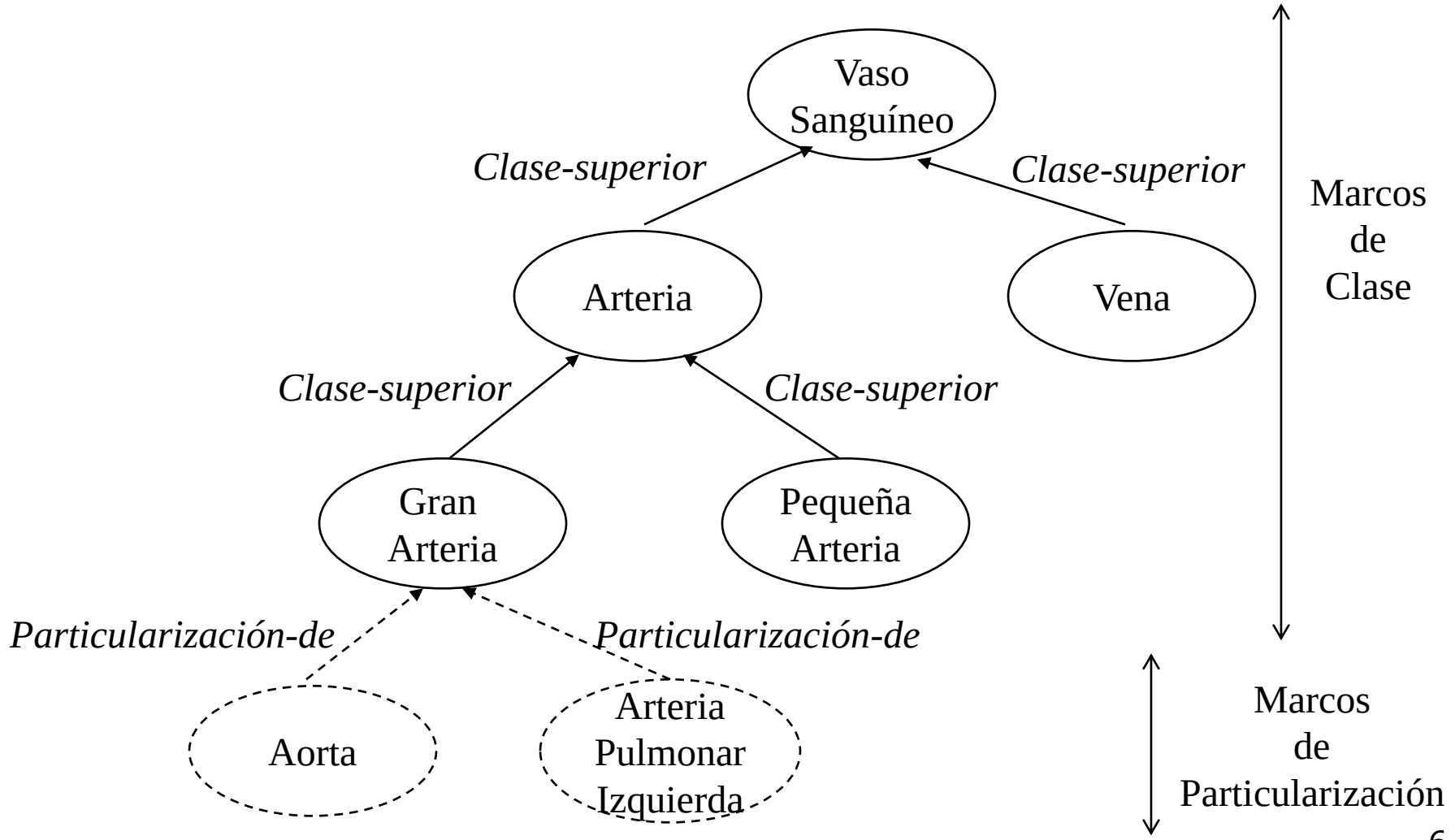
  conocimiento relativo a elementos individuales

- Relación clase-superior (MC×MC)

  relación de orden parcial: reflexiva, antisimétrica, transitiva (*inclusión*)
- Relación particularización-de (MP $\rightarrow$ MC)

  función total sobre MP (*pertenencia*)

Jerarquía Marcos



Estructura de un marco

- Según modelo básico

  Nombre

  Relación: clase-superior o particularización-de

  Atributos (slots)

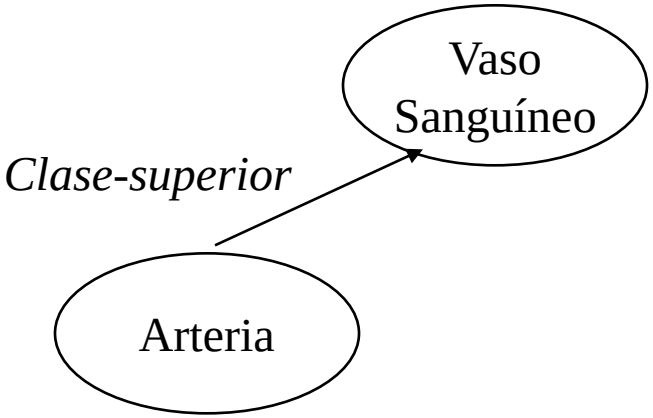
- Alternativa

  Nombre

  Atributos

  - de relación: clase-superior, particularización-de
  - de propiedades

Ejemplo marcos



Clase vaso-sanguíneo **es clase-superior** T;
contiene= fluido-sanguíneo;
forma = tubular
end

Clase arteria **es clase-superior** vaso-sanguíneo;
sangre = rica-oxigeno
end

Sintaxis Marcos (Jerarquía de Herencia: Árbol)

<marco> ::= <clase> | <particularización>

<clase> ::= **clase** <nombre-clase> **es**
clase-superior <espec-clase-superior> ;
<atributos-clase>
end

<espec-clase-superior> ::= <nombre-clase> | **T**

<atributos-clase> ::= <declaración> {;<declaración> }* | ε

<declaración> ::= <par-atributo-tipo> | <par-atributo-valor>

<particularización> ::= **particularización** <nombre-particularización> **es**
particularización-de <nombre-clase> ;
<atributos-particularización>
end

<atributos-particularización> ::= <par-atributo-valor> {;<par-atributo-valor>}* | ε

<par-atributo-tipo> ::= <nombre-atributo> : <tipo>

<par-atributo-valor> ::= <nombre-atributo> = <valor>

<tipo> ::= **int** | **real** | **string** | <enumerado> | <nombre-clase>

<enumerado> ::= **{** <valor> {,<valor>}* **}**

<valor> ::= <constante-elemental> | <nombre-particularización>

Restricciones

- Nombre único

  restricción de unicidad sobre <nombre-clase>, <nombre-particularización>

- Las constantes elementales

  han de pertenecer a alguno de los tipos predefinidos

- El valor <nombre-particularización>

  ha de ser el nombre de un marco de particularización del sistema, especialización de la clase del tipo declarado

Restricciones modelo básico

- Tipado

  cada <par-atributo-valor> que ocurra en una instancia

  ha de estar declarado en alguna de sus generalizaciones

  ha de pertenecer al tipo declarado

- Sin excepciones

  cada <par-atributo-valor> o <par-atributo-tipo> ocurre una sola vez en cada camino de la jerarquía

Ejemplo representación

- La arteria pulmonar izquierda tiene un diámetro aproximado de 0,4 cm., está situada en la parte superior del brazo y contiene sangre rica en oxigeno

particularización arteria-pulmonar-izquierda **es**
particularización-de arteria;
diámetro = 0.4;
situación = brazo;
sangre = rica-oxigeno
end

particularización brazo **es**
particularización-de extremidad;
posición=superior
end

Herencia simple sin excepciones

- Simple
  la jerarquía de herencia tiene estructura de árbol
- Sin excepciones
  cada <par-atributo-valor> o <par-atributo-tipo> ocurre una sola vez en cada camino de la jerarquía
- Simplificación adicional
  prescindimos de la información de tipo

Herencia simple sin excepciones

Regla de inferencia que consiste en asociar a cada marco todos los atributos de sus generalizaciones en la jerarquía de herencia

- Atributos: específicos / heredados

Ejemplo herencia simple sin excepciones

Clase vaso-sanguíneo **es clase-superior** T;
contiene= fluido-sanguíneo; forma = tubular
end

Clase arteria **es clase-superior** vaso-sanguíneo;
sangre = rica-oxigeno
end

Particularización aorta **es particularización-de** arteria;
diámetro = 2,5
end

- Marco aorta
  atrib. Específicos: diámetro
  atrib. Heredados: contiene, forma, sangre

Descripción recursiva herencia simple

Procedimiento Hereda(nombre-marco, pares-atributo-valor)
if nombre-marco=T **then**
return(pares-atributo-valor)
end;
pares-atributo-valor $\leftarrow$ pares-atributo-valor $\cup$
Atributos(nombre-marco);
return(Hereda(ClaseSuperior(nombre-marco), pares-atributo-valor))
end

Relación Marcos / LPO

- Limitado a Herencia simple sin excepciones ni información de tipo
- Directrices:
  - Nombre particularizaciones: símbolo constante
  - Nombre clases: símbolo predicado unario
  - particularización-de: predicado unario (clase), término constante (particularización)
  - clase-superior: implicación lógica (salvo T) y cuantificación universal
  - nombre atributo: símbolo función
  - par atributo-valor: igualdad entre término y constante (der. implicación si clase)
- Axiomas: igualdad + nombre único (nombre marcos y constantes)

Esquema general cambio representación

Clase C **es clase-superior** S;
$a_1=b_1$; $a_2=b_2$; ... $a_n=b_n$
end

$\forall x\ (C(x) \supset S(x))$
$\forall x\ (C(x) \supset a_1(x)=b_1)$
$\forall x\ (C(x) \supset a_2(x)=b_2)$
.
.
.
$\forall x\ (C(x) \supset a_n(x)=b_n)$

Particularización I **es particularización-de** C;
$j_1=k_1$; $j_2=k_2$; ... $j_m=k_m$
end

$C(I)$
$j_1(I)=k_1$
$j_2(I)=k_2$
.
.
.
$j_m(I)=k_m$

Restricción ocurrencia única par atributo-valor:
Garantiza existencia de un modelo

Ejemplo Herencia simple sin excepciones

Particularización Aorta

I ARTERIA(Aorta)

II diámetro(Aorta) =2,5

Particularización aorta **es**
particularización-de arteria;
diámetro = 2,5
end

Clase arteria **es**
clase-superior vaso-sanguíneo;
sangre = rica-oxigeno
end

Clase Arteria

III $\forall x$ (ARTERIA(x) $\supset$ VASO-SANGUINEO(x))

IV $\forall x$ (ARTERIA(x) $\supset$ sangre(x)=rica-oxigeno)

Clase vaso-sanguíneo **es**
clase-superior T;
contiene= fluido-sanguíneo;
forma = tubular
end

Clase Vaso-sanguíneo

V $\forall x$ (VASO-SANGUINEO(x) $\supset$ contiene(x)=fluido-sanguíneo)

VI $\forall x$ (VASO-SANGUINEO(x) $\supset$ forma(x)=tubular)

Ejemplo derivación

- De I y III, aplicando IU (sobre III) y MP al resultado (con I):

VII VASO-SANGUINEO(Aorta)

- De VII y V, aplicando IU (sobre V) y MP al resultado (con VII)

VIII contiene(Aorta)=fluido-sanguíneo

- De modo similar

IX sangre(Aorta)=rica-oxigeno

X forma(Aorta)=tubular

Herencia simple con excepciones

- Se permiten múltiples ocurrencias de un par atributo-valor en un camino de herencia
- Excepción: se produce una excepción cuando el valor de un atributo especifico de un marco es distinto del valor del mismo atributo en una de sus generalizaciones
  - valor excepcional: el del marco más específico
- Modificación de la herencia: se hereda el valor más específico
- Aporta: mayor flexibilidad diseño jerarquía, excepciones
- Consecuencia: ¿Semántica Herencia?

Ejemplo excepciones

Clase vaso-sanguíneo **es clase-superior** T;
    contiene= fluido-sanguíneo;
    forma = tubular
end

Clase arteria **es clase-superior** vaso-sanguíneo;
    sangre = rica-oxigeno
end

Particularización aorta **es particularización-de** arteria;
    diámetro = 2,5
end

Particularización arteria-pulmonar-izquierda
    es particularización-de arteria;
    diámetro = 0,4;
    sangre=pobre-oxigeno
end

Descripción r. herencia simple con excepciones

Procedimiento Hereda(nombre-marco, pares-atributo-valor)
if nombre-marco=T **then**
return(pares-atributo-valor)
end;
pares <--- Atributos(nombre-marco)
pares-atributo-valor ← pares-atributo-valor ∪
NuevosAtributos(pares, pares-atributo-valor);
return(Hereda(ClaseSuperior(nombre-marco), pares-atributo-valor))
end

Interés de la herencia simple con excepciones

- Permite razonamiento por defecto, que no se puede abordar mediante lógica clásica:

  Si no tenemos el historial clínico de una persona, "suponer que tiene el corazón a la izquierda"

- Semántica
  - Semántica operacional
  - No es posible representarla mediante lógica clásica
    - Con el método propuesto para el caso de herencia simple con excepciones, se obtiene un conjunto de fórmulas inconsistentes
    - ¿Por qué?
  - Se puede formalizar con lógicas no monotónicas, para las que no existen métodos efectivos de derivación de carácter general

Extensión de la representación: FACETS

- Facet: define propiedades adicionales de los valores de los atributos

- Tipos principales: restricción, demon
- Restricción: limitaciones sobre los valores
  - value: valor actual del atributo
  - default: valor por defecto
- Demon: procedimiento invocado automáticamente por el sistema de marcos cuando se satisface alguna condición sobre el valor de un atributo
  - if-needed — se necesita un valor, pero no existe valor actual
  - if-added — se proporciona un valor al facet value
  - if-removed — se elimina un valor del facet value

Sintaxis Marcos
(Herencia simple con excepciones, facets, no tipada)

<marco> ::= <clase> | <particularización>
<clase> ::= **clase** <nombre-clase> **es**
clase-superior <espec-clase-superior> <atributos>
end
<espec-clase-superior> ::= <nombre-clase> | **T**
<particularización> ::= **particularización** <nombre-particularización> **es**
particularización-de <nombre-clase> <atributos>
end
<atributos> ::= {**;**<par-atributo-facet>}* | ε
<par-atributo-facet> ::= <nombre-atributo> **= (**<facet> {**,**<facet> }***)**
<facet> ::= <nombre-facet> <valor> |
demon <tipo-demon> <llamada-demon>
<nombre-facet> ::= **value** | **default**
<tipo-demon> ::= **if-needed** | **if-added** | **if-removed**
<llamada-demon> ::= <nombre-procedimiento> **(**<arg> {, <arg>}* | ε **)**
<arg> ::= a:<nombre-atributo> | <valor>
<valor> ::= <constante-elemental> | <nombre-particularización>

Ejemplo: facets

Clase vaso-sanguíneo **es clase-superior** T;
    contiene = (**default** fluido-sanguíneo);
    forma = (**default** tubular)
end

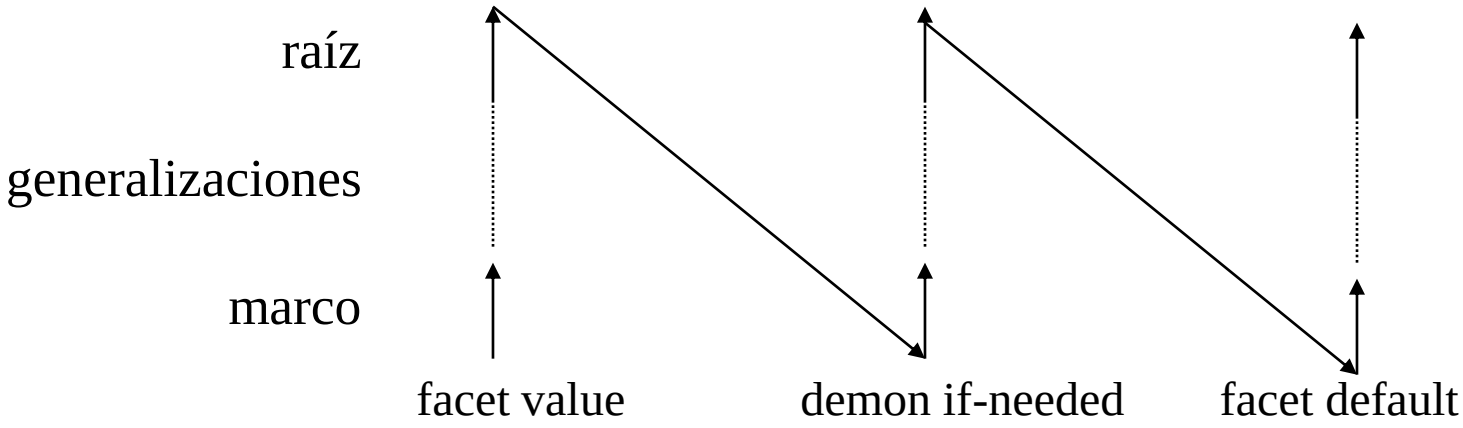
Clase arteria **es clase-superior** vaso-sanguíneo;
    sangre = (**default** rica-oxigeno);
    presión-sanguínea = (**default** 20);
    flujo-sanguíneo = (**default** 4);
    resistencia = (**demon if-needed** R(a:presión-sanguínea, a:flujo-sanguíneo),
                default 8)
end

Particularización aorta **es particularización-de** arteria;
    diámetro = (**value** 2,5)
end

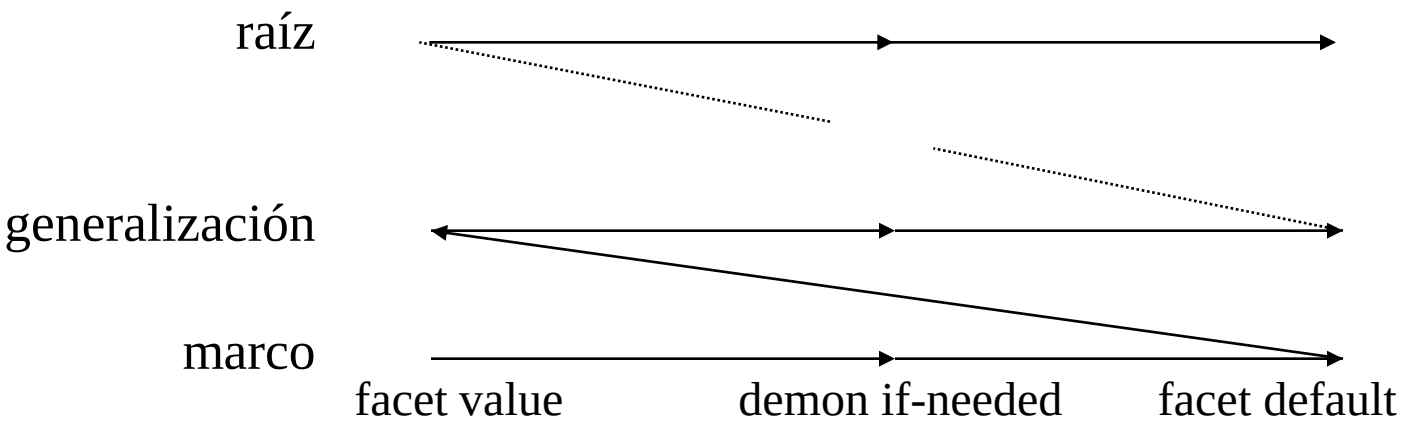
Particularización arteria-pulmonar-izquierda **es particularización-de** arteria;
    diámetro = (**value** 0.4);
    sangre = (**value** pobre-oxigeno)
    resistencia = (**default** 12)
end

Facets y Herencia: Herencia N, Z

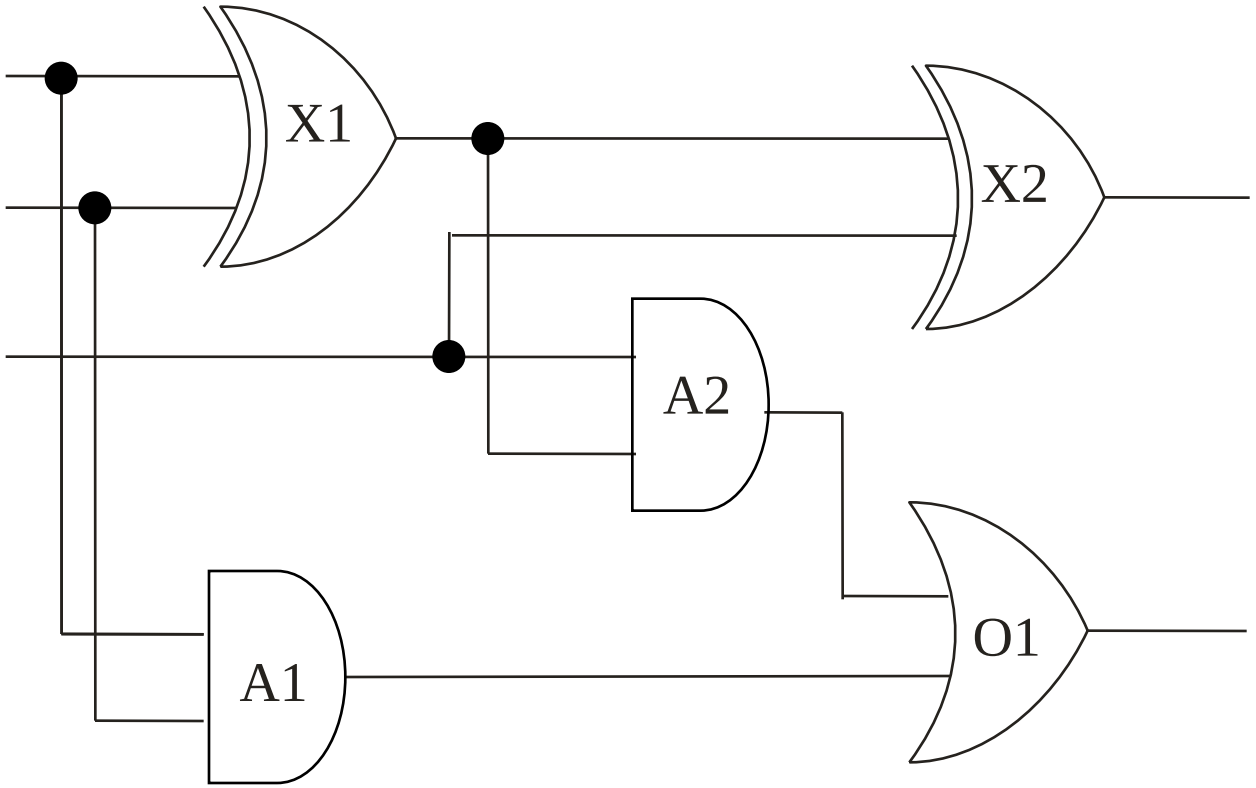
- Herencia N



- Herencia Z



Ejemplo: sumador completo
Tarea: simulación



ONTOLOGÍA sistema de marcos

- Herencia simple con excepciones, facets, Z
- Representar
  - puertas:
    - Tipos; clases puerta, puertaAND, puertaOR,…
    - Puertas específicas; particularizaciones A1, O1,…
  - Terminales de las puertas
    - Atributos: en1, en2, sal
  - Señales
    - Los atributos toman valores: 0,1 (no se puede representar sin información de tipo)
  - Entrada/salida circuito
    - Clase vectorEntrada, vectorSalida
    - Valores específicos: particularizaciones entradaCircuito, salidaCircuito
  - Conexiones:
    - Implícitas
    - En cada puerta, en dirección causal, indicando conexión de la entrada
    - En la salida del circuito, en dirección causal

Herencia simple con excepciones, facets, Z

- Reglas generales dominio
  - Funcionamiento de las puertas
    - Demon if-needed en las salidas
    - Funciones and, or, xor
    - Ejemplo. Para inferir el valor del atributo sal de una puerta AND: *sal=(**demon if-needed** and(a:en1, a:en2)*
  - Funcionamiento de las conexiones
    - Igualar los valores de los terminales conectados
    - En dirección causal
    - demon if-needed
    - Funciones obtener1, obtener2, obtener3, obtenerSal:

***Funcion** obtener1(?X)*
Devolver(?X.en1)
End funcion

***Funcion** obtenerSal(?X)*
Devolver(?X.sal)
End funcion

Ejemplo. Para inferir el valor del atributo en2 de la puerta O1:

*en2 = (**demon if-need** obtenerSal(A1))*

Ejemplo, herencia Z

Clase vectorEntrada
    es clase-superior T;
    en1 = (**default** 0);
    en2 = (**default** 0);
    en3 = (**default** 0)
end

Particularización
    entradaCircuito
    es particularización-de
    vectorEntrada
    en1= (value 1);
    en2 = (vaue 1);
    en3= (value 0)
end

Clase vectorSalida
        es clase-superior T;
    sal1 = (**default** 0);
    sal2 = (**default** 0);
end

Particularización salidaCircuito
        es particularización-de
    vectorSalida
    sal1 = (**demon if-needed**
    obtenerSal(X2));
    sal2 = (**demon if-needed**
    obtenerSal(O1))
end

Ejemplo, herencia Z

Clase puerta **es clase-superior** T;
    en1 = (**default** 0); en2 = (**default** 0) ; sal = (**default** 0)
end

Clase puertaAND **es clase-superior** puerta;
    sal = (**demon if-needed** and(a:en1, a:en2)
end

Particularización A1
        es particularización-de puertaAND;
    en1 = (**demon if-need** obtener1(entradaCircuito));
    en2 = (**demon if-need** obtener2(entradaCircuito))
end

Clase puertaOR **es clase-superior** puerta;
    sal = (**demon if-needed** or(a:en1, a:en2)
end

Particularización O1
        es particularización-de puertaOR;
    en1 = (**demon if-need** obtenerSal(A2));
    en2 = (**demon if-need** obtenerSal(A1))
end

Ejercicio Ontología (I)

- Completar la ontología del sumador completo para la tarea de simulación usando los marcos como LRC, con:
  - Herencia simple con excepciones
  - Facets
  - Herencia Z

Integración representaciones

- Marcos y Reglas
  - Sistema de producción
    - La jerarquía de marcos estructura la memoria de trabajo
  - Sistemas de marcos
    - Clase reglas, que se invocan vía demon

- A titulo ilustrativo, sistema de producción con memoria de trabajo estructurada mediante marcos

Sistemas de producción

- Lenguaje O-A-V

  O ::= <nombre-clase> | <nombre-particularización> | <variable>

  A ::=<nombre-atributo>

  V ::=<constante-elemental> | <nombre-particularización> | <variable>

- <variable>
  - palabra que empieza por ?
  - se puede instanciar a: <nombre-clase> | <nombre-particularización> | <constante-elemental>

- Condiciones y acciones

  <condición> ::= <predicado> (O, A, V)

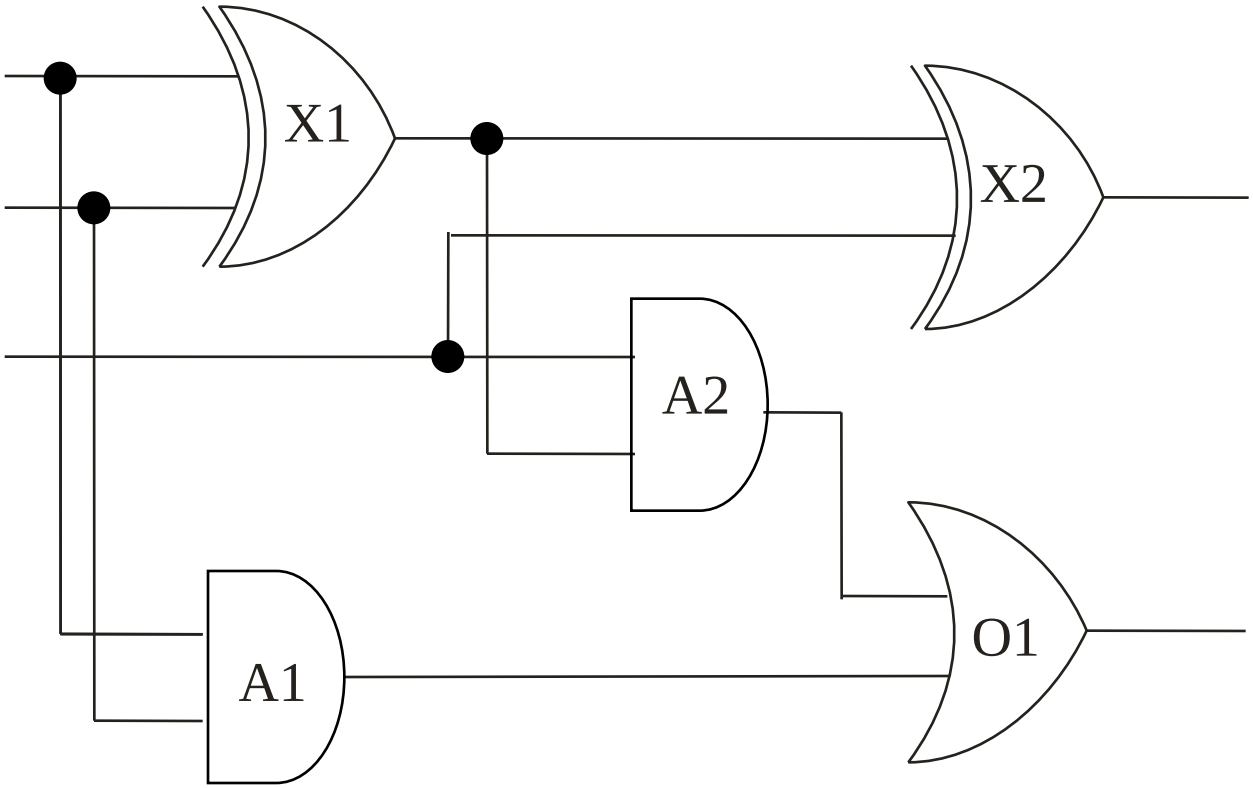
  <conclusión> ::= <acción>(O, A, V | =llamada-función)

- Marcos: herencia simple con excepciones, facets, Z

Modificaciones sistema de marcos

- Prioridad en la inferencia de valores:
  1. facet **value**
  2. facet **if-needed**
  3. encadenamiento de reglas
  4. facet **default**

Ejemplo: sumador completo
Tarea: simulación



Ontología sistema de producción con marcos

- Similar sistema de marcos
- Comportamiento puertas: reglas

Ejemplo, herencia Z

Clase vectorEntrada
    es clase-superior T;
    en1 = (**default** 0);
    en2 = (**default** 0);
    en3 = (**default** 0)
end

Particularización entradaCircuito
    es particularización-de
    vectorEntrada
    en1= (value 1);
    en2 = (vaue 1);
    en3= (value 0)
end

Clase vectorSalida
        es clase-superior T;
    sal1 = (**default** 0);
    sal2 = (**default** 0);
end

Particularización salidaCircuito
        es particularización-de
    vectorSalida
    sal1 = (**demon if-needed**
    obtenerSal(X2));
    sal2 = (**demon if-needed**
    obtenerSal(O1))
end

Ejemplo, herencia Z

```
Clase puerta es clase-superior T;
    en1 = (default 0);
    en2 = (default 0);
    sal = (default 0)
end

Clase puertaAND es clase-superior puerta;
end

Particularización AND1 es particularización-de puertaAND
    en1 = (demon if-needed obtener1(entradaCircuito))
    en2 = (demon if-needed obtener2(entradaCircuito))
end

Particularización OR1 es particularización-de puertaOR
    en1 = (demon if-needed obtenerSal(A2))
    en2 = (demon if-needed obtenerSal(A1))
end
```

Modelado de comportamiento mediante reglas

if iguales(?x, particularización-de, puertaAND) **and**
iguales(?x, en1, ?in1) **and**
iguales(?x, en2, ?in2)
then añadir(?x, sal, =and(?in1, in2?)) **fi**

Atención: requiere encadenamiento hacia atrás

(por representación elegida en el sistema de marcos)

Ejercicio Ontología (II)

- Completar la ontología del sumador completo para la tarea de simulación usando como LRC un sistema de producción con la MT estructurada mediante marcos, con:
  - Herencia simple con excepciones
  - Facets
  - Herencia Z