Representación del conocimiento

Lógica y representación del conocimiento.

Contenidos

1. Papel de la lógica en la representación del conocimiento.
2. Principios de Ingeniería de Conocimiento en Lógica de Primer Orden.
3. Ejemplo de desarrollo de una ontología específica y base de conocimiento en el dominio de los circuitos digitales.

1. Papel de la lógica en la representación del conocimiento

1. Papel de la lógica en la representación del conocimiento

- Intuitivamente atractiva como LRC:
  - Sintaxis bien definida.
  - Semántica precisa.
  - Mecanismos formales de deducción.

- Dificultades:
  - Problema de la cualificación.
  - Dinámica.
  - Incertidumbre.
  - Creencias.

Problema de la cualificación

- Limitación de las sentencias cuantificadas universalmente

- $\forall x\ (Ave(x) \supset Vuela(x))$
  - Puede ser cierta en algunos dominios
  - Presenta numerosas excepciones

- $\forall x\ (Ave(x) \wedge \neg Pingüino(x) \supset Vuela(x))$
  - Solución *Ad hoc*
  - No generalizable

- Sin solución en lógica clásica (como LPO)

Debate histórico

- Escuela “Logicista”: Inteligencia artificial como lógica aplicada. (Nilson, McCarthy)
- Frente a: la lógica como elemento de análisis (Misky, Newell), pero no de resolución de problemas(complejos).
- Cuena [99]:
  - Permite formalizar y mecanizar el razonamiento deductivo.
  - Permite modelar el conocimiento de un agente con independencia de su implementación.

Tendencia actual

- Lógica como lenguaje de modelado del conocimiento.
- Desarrollo del concepto de **ontología** en el ámbito de la representación del conocimiento
  - “Sistema particular de categorías sistematizando cierta visión del mundo”. [Guarino 98]
  - “Teoría particular de la naturaleza del ser o de la existencia”. [Rusell, Norvig 2010]
  - “Una ontología es una especificación formal de una conceptualización compartida”. [Studer 98]
- Uso de la lógica para la creación de ontologías.

2. Principios de Ingeniería del Conocimiento en Lógica de Primer Orden

Ingeniería del conocimiento

- Proceso de construcción de una base de conocimiento.

- Requiere:
  - Adquisición del conocimiento.
  - **Formalización.**
  - Representación.
  - Implementación.

- En LPO
  - La formalización proporciona la representación y la implementación.

Ingeniero del conocimiento

- Profesional que desarrolla la base de conocimiento.
- Requiere:
  - Conocer suficientemente el dominio.
  - Conocer los lenguaje de representación.
  - Conocer los mecanismos de inferencia.
  - Aspectos de diseño e implementación.
- Conocer suficientemente el dominio.
  - Posible en dominios sencillos, pequeños.
  - En general, precisa la colaboración de expertos del dominio y proceso de adquisición.

Proceso de ingeniería del conocimiento en LPO

- Orientado al dominio y a la tarea (Ontología específica del dominio y de la tarea)
- Proceso iterativo:
  1. Identificar la tarea.
  2. Reunir el conocimiento relevante.
  3. Elaborar un vocabulario (a veces denominado, erróneamente, ontología).
  4. Codificar el conocimiento general del dominio.
  5. Codificar una descripción de una instancia específica del problema.
  6. Plantear preguntas al procedimiento de inferencia y obtener respuestas.
  7. Depurar la base de conocimiento.

1- Identificar la tarea

- Rango de cuestiones que la base de conocimiento debe soportar.
  - Ej. Asistente al diagnóstico (Poole & Mackworth)
    - ¿Qué bombillas lucen?
    - ¿Qué interruptor está estropeado?

- Tipos de hechos disponibles para describir instancias específicas del problema.
  - Ej. Asistente al diagnóstico (Poole & Mackworth)
    - Componentes.
    - Conexiones.
    - Tensión en los cables…

2- Reunir el conocimiento relevante

- Posibilidad de que el ingeniero del conocimiento sea un experto del dominio.
- En general, trabajar con un experto en un proceso de adquisición del conocimiento.
- Obtener una descripción informal que permita:
  - Entender como funciona el dominio.
  - Apreciar el alcance de la base de conocimiento.
    - Que elementos hay que representar.
- La dificultad aumenta con la complejidad del dominio de aplicación.

3- Elaborar un vocabulario u ontología

- Elaborar un vocabulario de nombres de predicados, funciones y constantes (el vocabulario de la Ontología).
- Especifica qué existe (individuos y relaciones).
- No especifica sus propiedades e interrelaciones, pero es clave para la posterior descripción de las mismas.
- Ej. Asistente al diagnóstico (Poole & Mackworth)
  - Elementos del dominio: nombres de constantes
    - bombillas, símbolos **constantes**: L1, L2
  - Propiedades, relaciones entre elementos del dominio: sombres de funciones o predicados:
    - Conexiones, símbolo **predicado**: Connected(L1,W0)

4- Codificar el conocimiento general del dominio

- Elaborar los axiomas del dominio.
- Proporciona las propiedades de los elementos de la Ontología.
  - En otras palabras, proporciona el significado de los símbolos de la Ontología.
- EJ. Asistente al diagnóstico (Poole & Mackworth)

  Funcionamiento bombillas:

  $$\forall x\ [(Light(x) \wedge OK(x)) \supset (Live(x) \supset Lit(x))]$$

  Funcionamiento conexiones:

  $$\forall x\ \forall y\ [(Connected(x,y) \wedge Live(y)) \supset Live(x)]$$

5- Codificar una descripción de una instancia específica del problema

- Si la ontología es adecuada, se limita a proporcionar un conjunto de sentencias atómicas sobre instancias de conceptos.

- Juegan un papel similar a los datos de entrada de una programa tradicional.
  - Proporcionados por sensores en un agente, por ejemplo.

- Ej. Asistente al diagnóstico (Poole & Mackworth)
  - Up(S3)
  - Live(OUTSIDE)
  - OK(L1)

6-Plantear preguntas al procedimiento de inferencia y …

- El procedimiento de inferencia utiliza los axiomas y los hechos específicos del problema para derivar los hechos que nos interesan.

- EJ. Asistente al diagnóstico (Poole & Mackworth)
  - ∃x Lit(x)
  - ∃x Live(x)
  - ∃x ∃y Connected(x,y)

7- Depurar la Base de Conocimiento

- Asumiendo que los procedimientos de inferencias son correctos, las respuestas obtenidas son “correctas para la base de conocimiento codificada”.

- Problemas de verificación:
  - Axiomas codificados de forma incorrecta.
  - Ausencia de axiomas.

- Problemas de validación
  - Los axiomas describen el conocimiento necesario.

7- Depurar la Base de Conocimiento

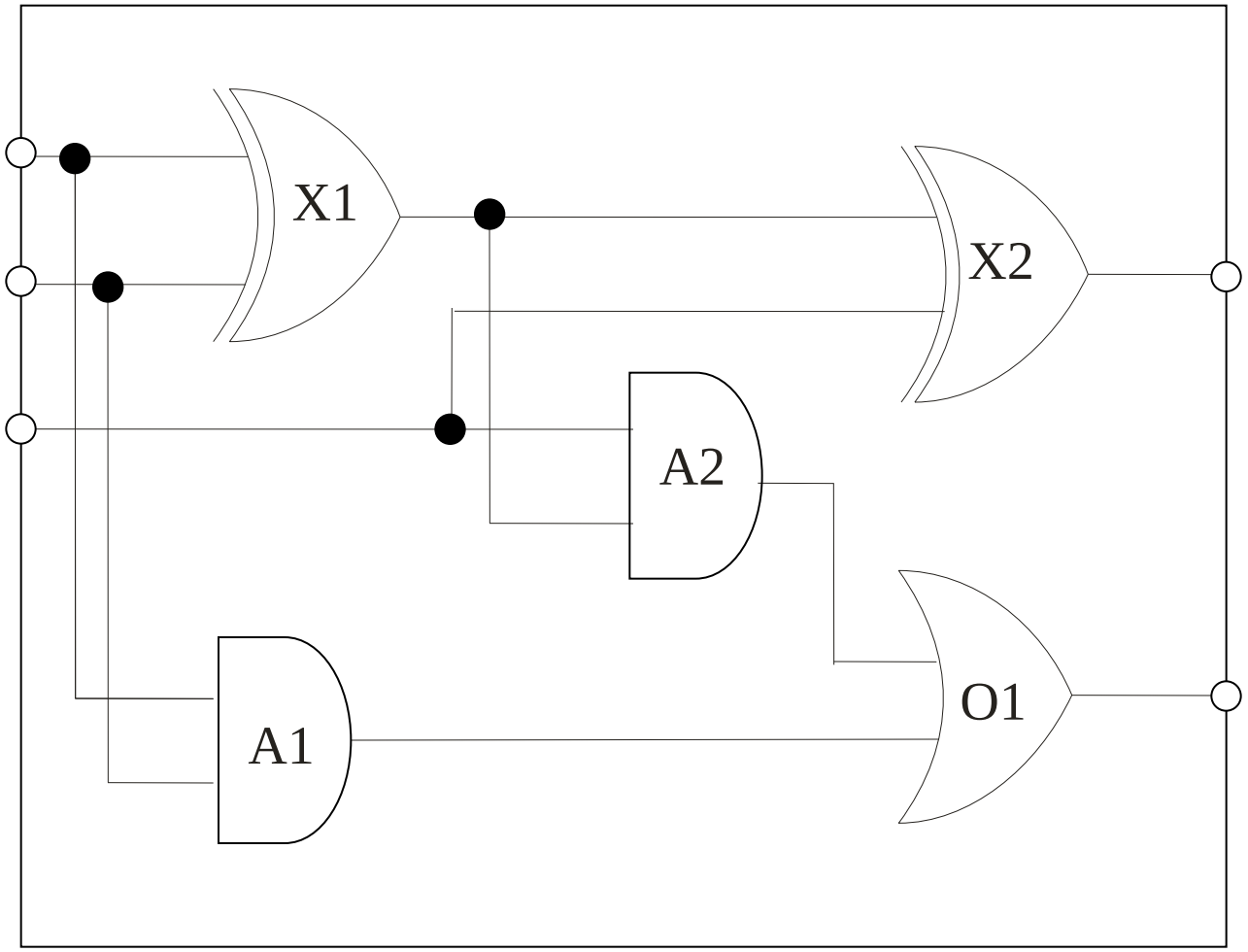
- EJ. Asistente al diagnóstico (Poole & Mackworth)
- Pregunta
  - Connected(L1,W0).
  - Respuesta: Si.
- Pero
  - Connected(W0, L1).
  - Respuesta: no.
- Plantearse si la representación es adecuada para el dominio y la tarea.

2. Ejemplo de desarrollo de ontología específica y base de conocimiento

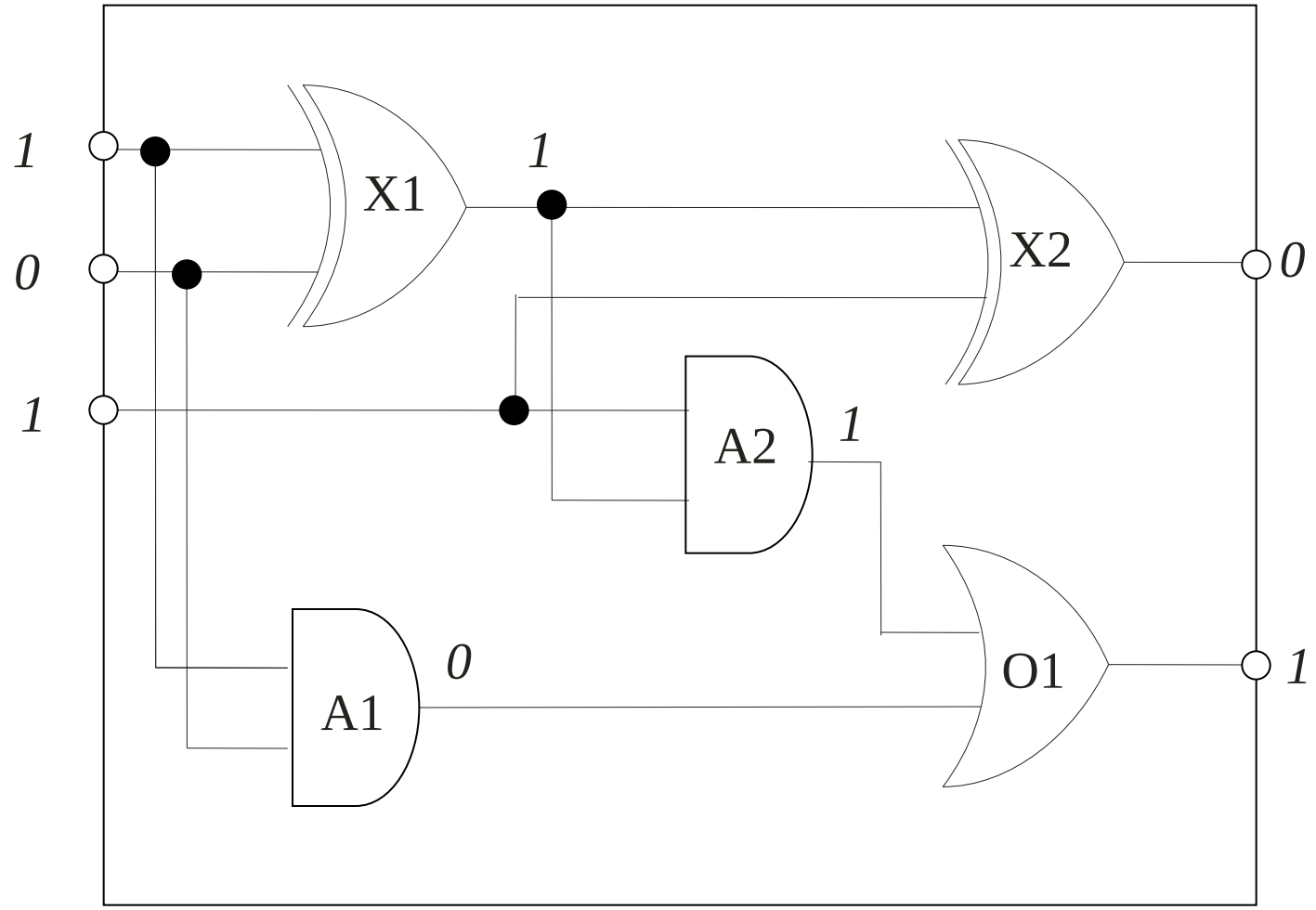
Ontología específica

- Dominio y tarea concreto.
- Mismo dominio pero distintas tareas requieren distintas ontologías específicas.
- No extrapolable a otros dominios.
- Quizás extensible a otras tareas.
- Dominio: circuitos electrónicos.
- Tarea: verificación.

Ejemplo circuito del dominio: Sumador completo de un bit con acarreo



Ejemplo circuito del dominio: Sumador completo de un bit con acarreo

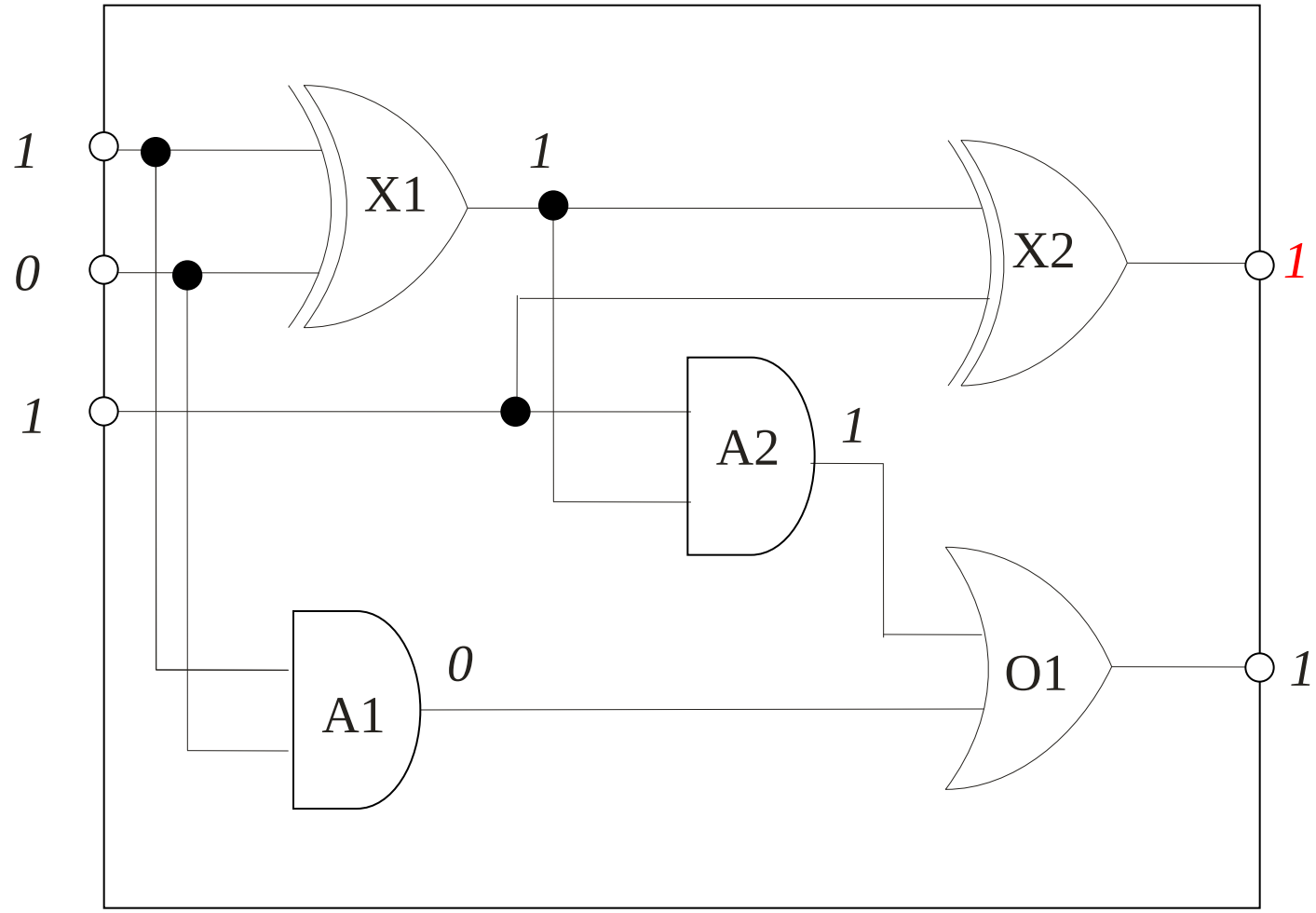


Tarea: verificación

- Verificación: comprobar que el circuito cumple con sus especificaciones de diseño.

- Para el sumador completo:
  - Para cualquier entrada, la salida corresponde a la suma de sus entradas.

Tarea: detección de fallos (monitorización)



1- Identificar la tarea

- La base de conocimiento debe permitir obtener las salidas para distintas combinaciones de entrada.
- También las entradas capaces de generar las salidas.
- Suficiente con conocer el nivel lógico de las señales.
- Posible pregunta:
  - ¿Valor de la señal en el terminal de salida 1 del circuito?
- Posible hecho:
  - A1 es una puerta AND
  - La señal en el terminal de entrada 2 es 0
  - El terminal de salida de la puerta X1 está conectado al terminal de entrada 1 de la puerta X2

2- Reunir el conocimiento relevante

- ¿Qué sabemos de los circuitos digitales?
  - Compuestos de cables y puertas
  - Las señales se propagan por los cables
  - Las puertas generan la señal de salida a partir de las entradas
  - Las puertas funcionan de distinta manera y tenemos 4 tipos de puertas, aunque solo tres en el ejemplo: AND, OR, XOR
  - Para conocer la salida, solo necesitamos la conectividad y la funcionalidad de las puertas (pero no la disposición del cableado)
- ¿Qué necesitamos representar?
  - Circuito, terminales, señales, puertas individuales, tipos de puertas, conectividad, funcionamiento de las puertas.

3- Elaborar un vocabulario u ontología

- Elaborar un vocabulario de nombres de predicados, funciones y constantes (Ontología).
- Especifica qué existe (individuos y relaciones).
- No especifica sus propiedades e interrelaciones, pero es clave para la posterior descripción de las mismas.

Vocabulario: referencias a las puertas y al circuito

- Nombre de las puertas, constantes:
  - X1, X2, A1, A2, O1
- Nombre del circuito, constante:
  - CS

Vocabulario: tipo de las puertas (I)

- Tipo de las puertas
  a) Predicados de tipo unario: XOR(X1)
  b) Predicados de tipo binario: TIPO(X1, XOR)
  c) Función unaria: tipo(X1)=XOR

- Las tres variantes permiten describir el tipo de las puertas, pero hay diferencias importantes:
  a) NO permite responder a la pregunta ¿De qué tipo es la puerta X1?
  b) Dificulta expresar que una puerta solo puede ser de un tipo.
  c) La semántica de las funciones garantiza que una puerta solo puede ser de un tipo.

Vocabulario: tipo de las puertas (II)

- Tipo de las puertas
  - Función unaria: tipo(X1)=XOR

- Requiere introducir constantes para referenciar el tipo de las puertas:
  - AND, OR, XOR, NOT

Vocabulario: referencias a terminales (I)

- Referencias a terminales
  a) Constantes: X1IN1, X1IN2, X1OUT1, CSOUT2,….
  b) Funciones: in(1,X1), in(2, X1), out(1,X1), out(2, CS)…

- Ambas opciones pueden ser válidas, pero:
  a) X1IN1 no guarda ninguna relación con X1: necesitamos sentencias adicionales para expresar que X1IN1 es el terminal de entrada 1 de X1
  b) in(1,X1) hace explícita la relación entre el terminal, designado por in(1,X1) y la puerta X1

Vocabulario: referencias a terminales (II)

- Referencias a terminales
  - Funciones binarias para terminales de puertas y circuitos: in(1,X1), in(2, X1), out(1,X1), out(2, CS)…
- Requiere constantes adicionales: 1, 2.

Vocabulario: señales (I)

- Dos opciones:
  a) Predicados ON, OFF: ON(in(1,X1)), OFF(out(1,CS))
  b) Función señal: señal(in(1,X1))=On, señal(out(1,CS))=Off

- Diferencias:
  a) Difícil preguntar ¿Cuales son los posibles valores de la señal en out(1,CS)?
  a) Sentencias adicionales para indicar que una señal solo puede tomar un valor
  b) Resuelve ambos problemas

Vocabulario: señales (II)

- Señales:
  - Función señal: señal(in(1,X1))=On, señal(out(1,CS))=Off

- Requiere introducir dos constantes:
  - On, Off

Vocabulario: conexiones

- Conexiones entre terminales: predicado binario
  - Conectado(in(1,CS), in(1, X1))

4- Codificar el conocimiento general del dominio

- Elaborar los axiomas del dominio
- Proporciona las propiedades de los elementos de la Ontología
- “Buena” Ontología
  - “Pocas” reglas generales
  - Claras y concisas

Conocimiento general: conexiones

- La conectividad es una propiedad conmutativa:

1 $\forall t_1 \, \forall t_2 \, (\text{Conectado}(t_1, t_2) \supset \text{Conectado}(t_2, t_1))$

- Los terminales conectados comparten la señal:

2 $\forall x_1 \, \forall x_2 \, (\text{Conectado}(x_1, x_2) \supset \text{señal}(x_1) = \text{señal}(x_2))$

Conocimiento general: señales (I)

- La señal en un terminal toma el valor On u Off:

$$\forall t_1 \; \forall t_2 \; (\text{señal}(t_1) = \text{On} \vee \text{señal}(t_1) = \text{Off})$$

- CUIDADO: la función *señal* garantiza que la señal en un terminal tiene un único valor pero ES POSIBLE INTERPRETAR On y Off con el mismo elemento del dominio
  - O bien: es posible que On=Off sea cierto

Conocimiento general: señales (II)

- La señal en un terminal toma el valor On u Off, pero no ambos:

a) $\forall t_1\ \forall t_2\ (\text{señal}(t_1)=\text{On} \vee \text{señal}(t_1)=\text{Off})$
$\forall t_1\ \forall t_2\ (\neg\text{señal}(t_1)=\text{On} \vee \neg\text{señal}(t_1)=\text{Off})$

b) $\forall t_1\ \forall t_2\ (\text{señal}(t_1) = \text{On} \vee \text{señal}(t_1) = \text{Off})$
$\text{On} \neq \text{Off}$

- Si vamos a utilizar expresiones señal(x) ≠ señal(y), la opción b) es preferible.

Conocimiento general: señales (III)

- La señal en un terminal toma el valor On u Off, pero no ambos:

3 $\forall t_1 \; \forall t_2 \; (\text{señal}(t_1) = \text{On} \vee \text{señal}(t_1) = \text{Off})$

4 $\text{On} \neq \text{Off}$

- Alternativa: **axioma de nombre único**.
  - Nombres diferentes designan entidades diferentes
  - Habitual en Programación Lógica (y en Prolog)

Conocimiento general: funcionamiento puertas. (AND I)

- Puerta AND

$\forall x$ [tipo(x)=AND $\supset$

señal(out(1, x))=On $\Leftrightarrow$ $\forall n$ señal(in(n, x))=On]

- Plantilla genérica para describir propiedades, comportamiento, etc., de elementos del domino:

“ Si un elemento del dominio es de un tipo ” $\supset$

“propiedades, modelo, etc. del tipo de elemento”

Conocimiento general: funcionamiento puertas. (AND II)

- Puerta AND

$$\forall x\ [tipo(x)=AND \supset$$
$$señal(out(1, x))=On \Leftrightarrow \forall n\ señal(in(n, x))=On]$$

- Para cualquier puerta AND permite derivar, sin utilizar más reglas generales del dominio:
  - Señal salida On si todas señales de entrada On
  - Señal de salida $\neg$ On si alguna señal de entrada $\neq$ On
  - Señales de entrada On si señal salida On

Conocimiento general: funcionamiento puertas. (AND III)

- Puerta AND

$\forall x$ [tipo(x)=AND $\supset$
señal(out(1, x))=On $\Leftrightarrow$ $\forall n$ señal(in(n, x))=On]

- Para cualquier puerta AND permite derivar, utilizando:

3 $\forall t_1 \; \forall t_2$ (señal(t_1) = On $\vee$ señal(t_1) = Off)

4 On $\neq$ Off

  - Señal salida Off si alguna señal de entrada Off
  - $\exists n$ señal(in(n, x))=Off si señal salida Off

Conocimiento general: funcionamiento puertas.

5 $\forall x$ [tipo(x)=AND $\supset$
señal(out(1, x))=On $\Leftrightarrow$ $\forall n$ señal(in(n, x))=On]

6 $\forall x$ [tipo(x)=OR $\supset$
...

7 $\forall x$ [tipo(x)=XOR $\supset$
...

8 $\forall x$ [tipo(x)=NOT$\supset$
...

Conocimiento general: recapitulación

- La elección del vocabulario permite representar todo el conocimiento general necesario para la tara de verificación con solo 8 FBFs
- La representación proporcionada está orientada a la verificación, pero NO introduce suposiciones adicionales relativas a “cómo” se realiza la tarea:
  - Por ejemplo, no se supone que sólo se razona en la dirección causal
- Fácilmente extensible a tareas que se realicen al mismo nivel de abstracción: detección de fallos, diagnosis…

5- Codificar instancia específica

- Proporcionar los elementos que describen una instancia específica.
- Con una buena Ontología, se limita a un conjunto de hechos que describen la configuración del problema actual.
- En nuestro ejemplo:
  - Indicar el tipo de las puertas.
  - Proporcionar las conexiones.

Codificar instancia específica

1 tipo(A1)=AND
2 tipo(A2)=AND
3 tipo(O1)=OR
4 tipo(X1)=XOR
5 tipo(X2)=XOR
6 Conectado(in(1,CS), in(1,X1))
7 Conectado(in(1,CS), in(1,A1))
8 Conectado(in(2,CS), in(2,X1))
9 Conectado(in(2,CS), in(2,A1))
10 Conectado(in(3,CS), in(2,X2))
11 Conectado(in(3,CS), in(1,A2))
...
16 Conectado(out(1,X2), out(1,CS)
17 Conectado(out(1,O1), out(2,CS))

6- Plantear preguntas al procedimiento de inferencia (I)

- ¿Qué combinación de entradas genera *Off* en salida 1 de C1 y *On* en salida 2 de C2?

$$\exists x \exists y \exists z \text{ señal}(\text{in}(1,\text{CS})) = x \wedge \text{señal}(\text{in}(2,\text{CS})) = y \wedge \text{señal}(\text{in}(3,\text{CS})) = z \wedge \text{señal}(\text{out}(1,\text{CS})) = \text{Off} \wedge \text{señal}(\text{out}(1,\text{CS})) = \text{On}$$

6- Plantear preguntas al procedimiento de inferencia (II)

- ¿Cuál es el conjunto de posibles valores en los terminales del circuito?

$$\exists x \exists y \exists z \exists u \exists z v \ \text{señal}(\text{in}(1,CS)) = x \wedge \text{señal}(\text{in}(2,CS)) = y \wedge \text{señal}(\text{in}(3,CS)) = z \wedge \text{señal}(\text{out}(1,CS)) = u \wedge \text{señal}(\text{out}(1,CS)) = v$$

La respuesta a esta pregunta genera la tabla de Entrada/Salida del dispositivo, lo que permite verificar su funcionamiento.

7- Depuración de la Base de Conocimiento

- Comportamiento anómalo típico:
  - Conocimiento general incompleto.

Suponer que omitimos la sentencia **3:**

$$\forall t_1 \; \forall t_2 \; (\text{señal}(t_1) = \text{On} \vee \text{señal}(t_1) = \text{Off})$$

Y que preguntamos por los valores de la salida para las entradas Off, Off, Off:

$$\exists x \exists y \; \text{señal}(\text{in}(1,\text{CS})) = \text{Off} \wedge \text{señal}(\text{in}(2,\text{CS})) = \text{Off} \wedge \text{señal}(\text{in}(3,\text{CS})) = \text{Off} \wedge \text{señal}(\text{out}(1,\text{CS})) = x \wedge \text{señal}(\text{out}(1,\text{CS})) = y$$

Depuración de la Base de Conocimiento

- El sistema no encuentra ninguna respuesta.
  - Por ejemplo, no puede derivar el valor de la señal a la salida de A1.

5 $\forall x$ [tipo(x)=AND $\supset$
señal(out(1, x))=On $\Leftrightarrow$ $\forall n$ señal(in(n, x))=On]

1. El sistema puede deducir *señal(in(1, A1))=Off* y *señal(in(2, A1))=Off*
2. La FBF **5** no permite derivar *señal(out(1, x))=On*
3. La FBF **5** permite derivar $\neg$*señal(out(1, x))=On*
4. No disponemos de la sentencia 3 para derivar *señal(out(1, x))=Off*

Recapitulación

- ¿Qué hemos hecho?
  - Representar, de forma declarativa, el conocimiento necesario para resolver el problema
- ¿Qué debemos evitar?
  - Proporcionar una solución que sólo sea útil para una instancia específica del problema

- Ejemplo sencillo de verificación de un circuito
- Sin más que describir otra instancia específica, aplicable a cualquier circuito combinacional
  - Propiedad de las representaciones declarativas
- Extensible a otros dominios