FUNDAMENTOS DE LA WEB SEMANTICA – 2023

UNIDAD 6: Lógica Desciptiva

Edelweis Rohrer InCo- Fing - Universidad de la República

Parte II – Razonamiento

Mecanismos de razonamiento

En la Parte I de esta unidad introduce la sintaxis y la semántica de una base de conocimiento formalizada usando *lógica descriptiva*. La semántica de lógica descriptiva se define en base al concepto de interpretación. Vimos que una interpretación que satisface los axiomas de una base de conocimiento es un *modelo* de la base de conocimiento, y que una *base de conocimiento es consistente si tiene al menos un modelo*.

La definición de modelo de una base de conocimiento introduce el *mecanismo de razonamiento* más básico que provee el formalismo de lógica descriptiva, la **consistencia**, que consiste en asegurar que el conjunto de axiomas de la base de conocimiento no contiene contradicciones. Existen también otros mecanismos de razonamiento que son útiles para inferir nuevo conocimiento a partir del conocimiento que está declarado en la base de conocimiento. Estos mecanismos son **satisfactibilidad**, **inclusión o subsumption** e **instanciación**.

El siguiente cuadro presenta la definición de los cuatro mecanismos de razonamiento principales.

Una base de conocimiento $\mathcal{K}$ es **consistente** si **existe un modelo I de $\mathcal{K}$**

Un concepto ***C*** **es satisfactible con respecto a** $\mathcal{K}$
si **existe un modelo I de $\mathcal{K}$ tal que** $C^{\mathrm{I}} \neq \emptyset$

Un concepto ***C*** **está incluido en** ***D*** **con respecto a** $\mathcal{K}$
si $C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$ **se satisface para todo modelo I de $\mathcal{K}$** Se denota: $\mathcal{K} \models C \sqsubseteq D$

Un individuo ***a*** **es una instancia de** un concepto ***C*** **con respecto a** $\mathcal{K}$
si $a^{\mathrm{I}} \in C^{\mathrm{I}}$ **para todo modelo I de $\mathcal{K}$** Se denota: $\mathcal{K} \models C(a)$

Consideremos la siguiente base de conocimiento $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

$\mathcal{T}$ = {*Mujer* ⊑ *Persona, Persona* ≡ *Mujer* ⊔ *Hombre*, *Madre* ≡ *Mujer* ⊓ ∃*tieneHijo.Persona*}

$\mathcal{A}$ = {*Mujer*(*maria*), *Mujer*(*juana*), *Hombre*(*diego*), *tieneHijo*(*maria*, *Diego*)}

Veamos si se cumple que $\mathcal{K} \models$ *Hombre* ⊑ *Persona* (*Hombre* está incluido en *Persona*)

Como *Persona* ≡ *Mujer* ⊔ *Hombre*, se va a cumplir que para todo modelo I de $\mathcal{K}$:

$Hombre^{\mathrm{I}} \sqsubseteq Persona^{\mathrm{I}}$, por lo que se cumple que *Hombre* está incluido en *Persona*.

Para ejercitar:

$\mathcal{K} \models$ *Madre*(*maria*)?

Ahora agreguemos a $\mathcal{T}$ los axiomas *Mujer* ⊓ *Hombre* ⊑ ⊥ y *C* ⊑ *Mujer* ⊓ *Hombre.*

Claramente el concepto *C* no es satisfactible, ya que no existe ninguna interpretación tal que $C^{I} \neq \emptyset$.

Algoritmo de Tableau

Dada una base de conocimiento $\mathcal{K}$= ($\mathcal{T}$, $\mathcal{A}$), donde $\mathcal{T}$ es el Tbox y $\mathcal{A}$ es el Abox, un *algoritmo de razonamiento* devuelve si la ontología es *consistente*, o sea si tiene un modelo. Además, hace *explícito conocimiento no declarado* en la ontología, aplicando los mecanismos de razonamiento de *satisfactibilidad*, *inclusión* e *instanciación* mencionados en la sección anterior.

Un *razonador* es una implementación de un algoritmo de razonamiento.

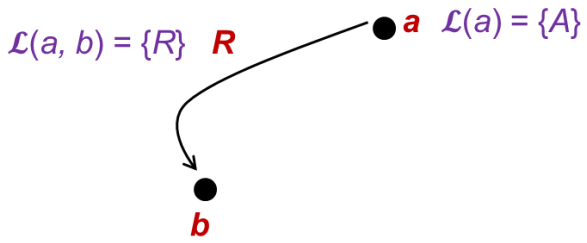
Existen diferentes algoritmos de razonamiento, entre ellos el algoritmo de *Tableau* es el más conocido, ya que aplica a todos los fragmentos de la familia de lógica descriptiva **[1]**. La mayoría de los razonadores, como Hermit y Pellet, implementan el algoritmo Tableau.

Dada una base de conocimiento $\mathcal{K}$, el algoritmo de Tableau intenta **construir un modelo de $\mathcal{K}$.** Si encuentra un modelo de $\mathcal{K}$**,** el algoritmo devuelve que $\mathcal{K}$ es **consistente.** Si no logra construir un modelo de $\mathcal{K}$**,** el algoritmo devuelve que $\mathcal{K}$ es **inconsistente.**

Para encontrar un modelo de la base de conocimiento $\mathcal{K}$, el algoritmo *construye una estructura de grafo o tableau* $\mathcal{L}$, que se compone de[1]:

- Un conjunto de **nodos etiquetados**, donde cada etiqueta es un nombre de *individuo* o de *variable* (nodos agregados por el algoritmo).
- Un conjunto de **arcos dirigidos entre nodos**
- Para todo **nodo** con etiqueta ***x***, un **conjunto de conceptos $\mathcal{L}$(*x*)**
- Para todo **par de nodos *x*, *y***, el **conjunto de roles $\mathcal{L}$(*x*, *y*)**

La siguiente figura ilustra un grafo donde *a*, *b* son individuos, *A* es un concepto y *R* es un rol.



El algoritmo de *Tableau* ejecuta **tres pasos** principales: *inicialización*, *aplicación de reglas* y *terminación*, que se describen a continuación.

Inicialización

En este paso los axiomas de la base de conocimiento $\mathcal{K}$ se transforman a la *forma normal de negación*, y a partir de los individuos de $\mathcal{K}$ y de los axiomas de Abox, se **construye un grafo o tableau inicial**.

La transformación a la forma normal de negación (FNN) se aplica a los conceptos y a los axiomas de Tbox de $\mathcal{K}$. En el caso de los conceptos, lo que se hace es mover la negación inmediatamente antes

[1] En este material se describe el grafo que se construye para lógicas como $\mathcal{ALC}$ o $\mathcal{ALCQ}$, para lógicas más expresivas el grafo construido tiene algunas complejidades adicionales que no forman parte del alcance de este curso.

de los nombres de conceptos, por ejemplo, el concepto *Abuela* ⊓ ¬*Madre* está en FNN, pero el concepto ¬(¬*Abuela* ⊔ *Madre*) no está en FNN.

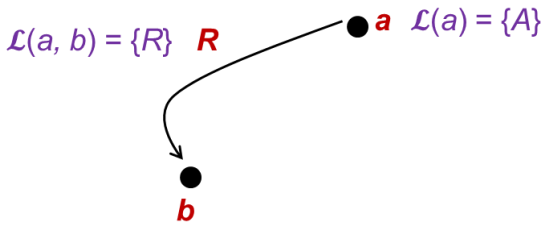
Además, los axiomas de Tbox $C \sqsubseteq D$ se transforman en conceptos $\neg C \sqcup D$, puesto que como se debe cumplir que todos los elementos que pertenecen a *C* también pertenezcan a *D*, para cualquier cualquier elemento del dominio se cumple que o bien no pertenece a *C* o pertenece a *D*.

Una vez que $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ se encuentra en FNN, el *grafo inicial* se construye siguiendo el procedimiento que se describe a continuación.

- Por cada individuo *a* en $\mathcal{K}$ se crea un nodo con etiqueta *a,* y se asigna $\mathcal{L}(a) = \emptyset$.
- Por cada par *a*, *b* de individuos, se crea un arco entre *a* y *b*, y se asigna $\mathcal{L}(a, b) = \emptyset$.
- Por cada axioma de $\mathcal{A}$ $C(a)$ en $\mathcal{K}$, se agrega el concepto *C* a $\mathcal{L}(a)$. Esto se denota $\mathcal{L}(a) \leftarrow C$.
- Por cada axioma de $\mathcal{A}$ $R(a, b)$ en $\mathcal{K}$, se agrega el rol *R* a $\mathcal{L}(a, b)$. Esto se denota $\mathcal{L}(a, b) \leftarrow R$.

Para la base de conocimiento $\mathcal{K} = \{C(a), \neg C \sqcap D(a)\}$, el grafo inicial tiene un único nodo *a* ($\mathcal{K}$ tiene un único individuo *a*), y además se asigna: $\mathcal{L}(a) = \{C, \neg C \sqcap D\}$

Para La base de conocimiento $\mathcal{K} = \{A \sqsubseteq B, A(a), R(a, b)\}$, se construye el grafo inicial que muestra la siguiente figura:



Aplicación de reglas

En este paso **se extiende el grafo** con la **aplicación de un conjunto de reglas de expansión**, que dependen de la lógica de la base de conocimiento $\mathcal{K}$, o sea de los constructores de los diferentes fragmentos de lógica descriptiva utilizados. Con la aplicación de las reglas, el grafo inicial se extiende: se agregan nodos, arcos, se asocian nuevos conceptos a los nodos y nuevos roles a los arcos del grafo.

Para las bases de conocimiento en la lógica $\mathcal{ALC}$, las reglas de expansión que se aplican son las que detallan a continuación.

- **Regla ⊓**: Si $C \sqcap D \in \mathcal{L}(x)$ y $\{C, D\} \not\subseteq \mathcal{L}(x)$ entonces se asigna $\mathcal{L}(x) \leftarrow \{C, D\}$
- **Regla ⊔**: Si $C \sqcup D \in \mathcal{L}(x)$ y $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ entonces se asigna $\mathcal{L}(x) \leftarrow \{C\}$ o se asigna $\mathcal{L}(x) \leftarrow \{D\}$
- **Regla ∃**: Si $\exists R.C \in \mathcal{L}(x)$ y no existe un nodo *y tal que* $R \in \mathcal{L}(x, y)$ y $C \in \mathcal{L}(y)$ entonces se realiza lo siguiente:
  - Se agrega un nodo con etiqueta *y*
  - Se asigna $\mathcal{L}(x, y) = \{R\}$
  - Se asigna $\mathcal{L}(y) = \{C\}$
- **Regla ∀**: Si $\forall R.C \in \mathcal{L}(x)$ y existe un nodo *y tal que* $R \in \mathcal{L}(x, y)$ y $C \notin \mathcal{L}(y)$ entonces se asigna $\mathcal{L}(y) \leftarrow \{C\}$
- **Regla $\mathcal{T}$**: Si un concepto $C \in \mathcal{T}$ (C corresponde a un axioma $C \sqsubseteq D$ transformado en $\neg C \sqcup D$) y $C \notin \mathcal{L}(x)$ entonces se asigna $\mathcal{L}(x) \leftarrow \{C\}$

Consideremos la siguiente base de conocimiento $\mathcal{K}$ donde se muestra al único axioma de Tbox ya transformado a FNN.

$\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

$\mathcal{T}$ = {*Humano* ⊑ ∃*tienePadre.Humano*} = {¬*Humano* ⊔ ∃*tienePadre.Humano*}

$\mathcal{A}$ = {*Ave*(*picaflor*), *Humano*(*picaflor*)}

El grafo inicial, que tiene un único nodo *picaflor* es el siguiente:

$\mathcal{L}$(*picaflor*) = {*Ave,* Humano}

La única regla que se puede aplicar es la Regla $\mathcal{T}$, quedando:

$\mathcal{L}$(*picaflor*) = {*Ave, Humano,* ¬*Humano* ⊔ ∃*tienePadre.Humano*}

Luego se aplica la Regla **⊔**. Si se elige ¬*Humano* el algoritmo detecta una contradicción, ya que el concepto *Humano* ya pertenece a $\mathcal{L}$(*picaflor*). Entonces el algoritmo de Tableau aplica un mecanismo llamado *backtracking*, que consiste en volver atrás y seleccionar el otro elemento de la disjunción, con lo que grafo queda:

$\mathcal{L}$(*picaflor*) = {*Ave, Humano,* ∃*tienePadre.Humano*}

En este punto es posible aplicar la Regla **∃.** Como no existe ningún nodo vinculado al nodo *picaflor* a través del rol *tienePadre*, lo que hace esta regla es crear un nuevo nodo *x*, vincular *picaflor* a *x* agregando un arco etiquetado con el rol *tienePadre*, y asociar el concepto *Humano* a *x*. El grafo queda de la siguiente manera:

$\mathcal{L}$(*picaflor*) = {*Ave, Humano,* ¬*Humano* ⊔ ∃*tienePadre.Humano*}

$\mathcal{L}$(*picaflor, x*) = {*tienePadre*}

$\mathcal{L}$(*x*) = {*Humano*}

Pero al agregarse un nuevo nodo x, a éste también se le debe aplicar las reglas $\mathcal{T}$ y **⊔,** quedando:

$\mathcal{L}$(*x*) = {*Humano,* ¬*Humano* ⊔ ∃*tienePadre.Humano*}, y luego:

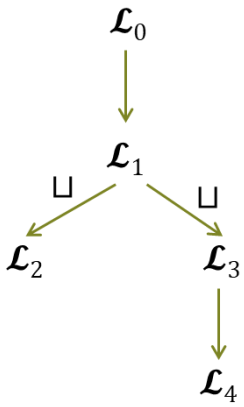
$\mathcal{L}$(*x*) = {*Humano,* ∃*tienePadre.Humano*}

Por lo tanto, se aplica la Regla **∃** y se crea un nuevo nodo, como se realizó para el nodo *picaflor*. Este procedimiento se aplicaría infinitamente, ya que el nuevo nodo *y* es idéntico al nodo *x*. Ante esta situación el algoritmo aplica un mecanismo de *bloqueo*, que consiste en no continuar aplicando reglas a los nodos *y* tales que $\mathcal{L}$(*y*) contiene el mismo conjunto de conceptos $\mathcal{L}$(*x*) que su antecesor *x.* El mecanismo de bloqueo se aplica para asegurar la terminación del algoritmo[2].

<u>Terminación</u>

El algoritmo de Tableau construye un grafo inicial $\mathcal{L}_0$, que con la aplicación de las reglas se va extendiendo, obteniéndose grafos $\mathcal{L}_1$, …, $\mathcal{L}_n$. En caso que se encuentre alguna contradicción, si se llegó a ella por la aplicación de alguna regla no determinística como la regla **⊔,** entonces, si aún le resta explorar otro camino, vuelve al grafo anterior y toma la otra alternativa. La siguiente figura ilustra este proceso.

[2] En lógicas más expresivas que $\mathcal{ALC}$, el mecanismo de bloqueo tiene una mayor complejidad, la cual no forma parte del alcance de este curso.



Cuando se obtiene un grafo sobre el que **no se pueden aplicar más reglas** y no se detectó **ninguna contradicción, el algoritmo de Tableau devuelve que $\mathcal{K}$** es **consistente.**

Cuando se obtiene un grafo sobre el que **no se pueden aplicar más reglas**, y por todos los caminos posibles se llega a una **contradicción, el algoritmo de Tableau devuelve que $\mathcal{K}$** es **inconsistente.**

La *estrategia de aplicación* de las reglas del algoritmo es **no determinística** por dos motivos:

- El *orden de aplicación de las reglas* no está preestablecido, es decir si en punto es posible aplicar dos reglas diferentes, el orden en que se apliquen es indistinto.
- Existen *reglas* que son en sí mismas *no determinísticas*, como la Regla **⊔.**

Algoritmo de Tableau y mundo abierto

Consideremos la siguiente base de conocimiento:

$\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

$\mathcal{T}$ = {*Hombre* ⊑ *Humano, Pájaro* ⊑ *Ave*}

$\mathcal{A}$ = {*Pájaro*(*miPicaflor*), *Hombre*(*Juan*)}

miPicaflor es *Humano*?

Juan es *Ave*?

Apliquemos el algoritmo de Tableau.

El grafo inicial es:

$\mathcal{L}$(*miPicaflor*) = {*Pájaro*}

$\mathcal{L}$(*Juan*) = {*Hombre*}

Aplicando la Regla $\mathcal{T}$:

$\mathcal{L}$(*miPicaflor*) = {*Pájaro, ¬Hombre* ⊔ *Humano, ¬Pájaro* ⊔ *Ave*}

$\mathcal{L}$(*Juan*) = { *Hombre, ¬Hombre* ⊔ *Humano, ¬Pájaro* ⊔ *Ave*}

Al aplicar la regla **⊔,** para el nodo *miPicaflor* el algoritmo puede elegir ir por *Humano* (pero podría elegir también ¬*Hombre*), y por Ave, ya que por ¬*Pájaro* se detecta una contradicción. Para el nodo *Juan*, elige *Humano* (por ¬*Hombre* hay contradicción), y *Ave* (aunque podría también elegir ¬*Pájaro*). El grafo queda:

$\mathcal{L}$(*miPicaflor*) = {*Pájaro, Humano, Ave*}

$\mathcal{L}$(*Juan*) = {*Hombre, Humano, Ave*}

En este punto ya no se puede aplicar más reglas, por lo que $\mathcal{K}$ es consistente. Se obtiene un modelo en el que tanto *miPicaflor* como *Juan* son ambos *Humano* y *Ave.* Por el paradigma de mundo abierto,

si no hay ningún axioma que establezca que *Humano* y *Ave* son disjuntos, no se puede asumir que lo sean.

Se agrega ahora a $\mathcal{T}$ el axioma *Ave* ⊓ *Humano* ⊑ ⊥ = ¬(*Ave* ⊓ *Humano*) ⊔ ⊥ = ¬*Ave* ⊔ ¬*Humano* ⊔ ⊥

Al aplicar la Regla $\mathcal{T}$ para este axioma, ¬*Ave* ⊔ ¬*Humano* ⊔ ⊥ se agrega a $\mathcal{L}$(*miPicaflor*) y a $\mathcal{L}$(*Juan*). ⊥ se descarta de entrada porque ningún elemento puede pertenecer al conjunto vacío. Si embrago, tanto ¬*Ave* como ¬*Humano* generan una contradicción en los nodos *miPicaflor* y *Juan,* ya que ambos tienen los conceptos *Humano* y *Ave.* Por lo tanto, al agregarse el concepto ¬*Ave* ⊔ ¬*Humano* ⊔ ⊥ a ambos nodos, la aplicación de la Regla ⊔ a los conceptos ¬*Hombre* ⊔ *Humano,* ¬*Pájaro* ⊔ *Ave*, debería haber seleccionado ¬*Hombre* para *miPicaflor* y ¬*Pájaro* para *Juan*. De esta forma no ocurre una contradicción, *miPicaflor* ya no clasifica como *Humano* y *Juan* ya no clasifica como *Ave*.

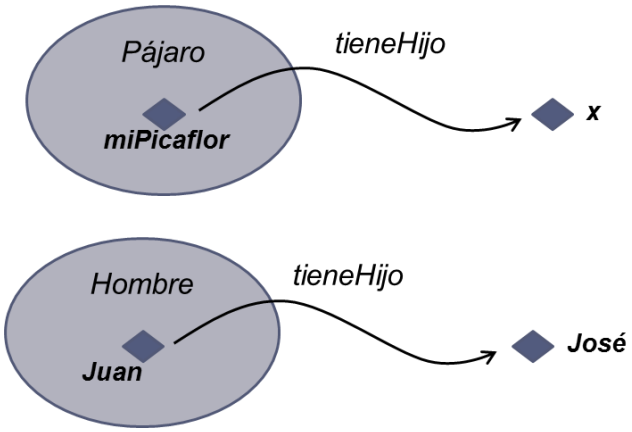
Es importante tener claro que <u>los axiomas de dominio y rango son usados por el razonador para hacer inferencias, no para generar una inconsistencia</u> en el caso que los individuos que están vinculados a través de un rol, no estén declarados como instancias del dominio de dicho rol (y lo mismo para el rango).

Consideremos la siguiente base de conocimiento:

$\mathcal{T}$ = {∃*tieneHijo.*⊤ ⊑ *Hombre*}

$\mathcal{A}$ = {*Pájaro*(*miPicaflor*), *Hombre*(*Juan*), *tieneHijo*(*Juan*, *José*), *tieneHijo*(*miPicaflor*, *x*)}

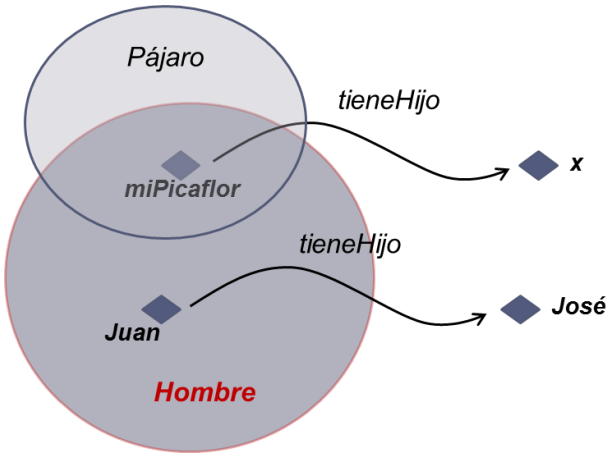
Gráficamente:



El axioma **∃*tieneHijo.*⊤ ⊑ *Hombre*** es un axioma de dominio porque establece que todos los elementos del dominio relacionados a otros por el rol *tieneHijo* deben pertenecer al concepto *Hombre*.

Sin entrar en el detalle de cada paso del algoritmo de Tableau aplicado a esta base de conocimiento, al aplicar la Regla $\mathcal{T}$ y agregar el concepto ¬∃*tieneHijo.*⊤ ⊔ *Hombre* a *miPicaflor*, el algoritmo se ve forzado a ir por *Hombre* (ya que por ¬∃*tieneHijo.*⊤ termina dándose una contradicción por el axioma *tieneHijo*(*miPicaflor*, *x*)), y entonces además de ser instancia de *Pájaro, miPicaflor* es también una instancia de *Hombre*, lo que es coherente con el axioma de dominio ∃*tieneHijo.*⊤ ⊑ *Hombre*, y no se detecta ninguna inconsistencia.

La siguiente figura ilustra el comportamiento del razonador.



Referencia:

[1] Description Logics Handbook: Theory, Implementation, and Applications. Editores Franz Baader, Diego Calvanese,Deborah L. McGuinness,Daniele Nardi y Peter F. Patel-Schneider. 2003.

Capítulos 1 y 2.