Tableaux para chequear consistencia de una base de conocimientos general en $\mathcal{ALC}$

Paula Severi

University of Leicester

Curso Noviembre-Diciembre 2018. Facultad de Ingeniería.
Universidad de la República, Montevideo, Uruguay.

Resumen

1. Problemas de inferencia para bases de conocimiento.
2. Tableaux para chequear consistencia de una Tbox simple.
3. Tableaux para chequear consistencia de una Tbox simple junto con una Abox.
4. Tableaux para chequear consistencia de una base de conocimientos general en $\mathcal{ALC}$.
5. Correctitud del Algoritmo.

Consistencia de una base de conocimiento en $\mathcal{ALC}$

Sea $\mathcal{I}$ una intepretación y $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ una base de conocimiendo.

$\mathcal{I}$ es un modelo de $\mathcal{T}$ (o $\mathcal{I}$ satisface $\mathcal{T}$)

si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ para todo $C \sqsubseteq D$ en $\mathcal{T}$.
Notación: $\mathcal{I} \models \mathcal{T}$.

$\mathcal{I}$ es un modelo de $\mathcal{K}$ (o $\mathcal{I}$ satisface $\mathcal{K}$)

1. $\mathcal{I}$ es modelo de $\mathcal{T}$
2. $a^{\mathcal{I}} \in C^{\mathcal{I}}$ para todo $a : C$ en $\mathcal{A}$
3. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ si $\langle a, b \rangle : R$ en $\mathcal{A}$.

Notación: $\mathcal{I} \models \mathcal{K}$.

Base consistente

$\mathcal{K}$ es consistente si tiene un modelo.

Algoritmo - Consistencia de una base en $\mathcal{ALC}$

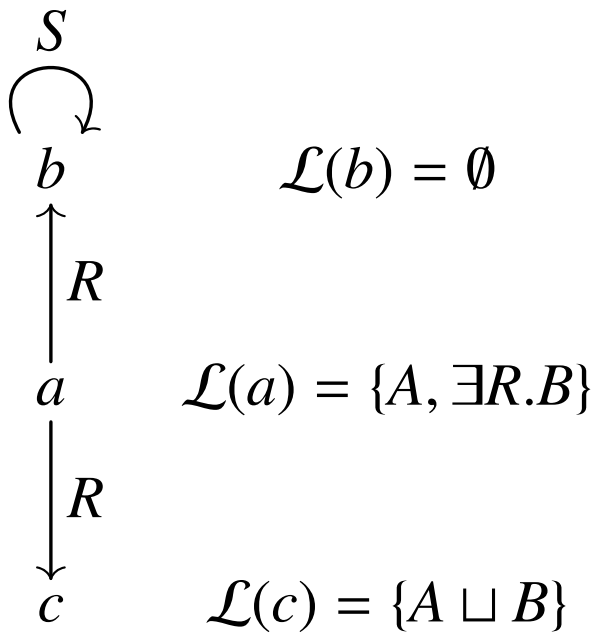
Inicialización del grafo a partir de una Abox

Dada una base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ en FNN, el tableau inicial para $\mathcal{K}$ se define:

1. Por cada individuo a en $\mathcal{K}$, creamos un nodo con etiqueta a y ponemos $\mathcal{L}(a) = \emptyset$.
2. Por cada par a, b de individuos, ponemos $\mathcal{L}(a, b) = \emptyset$.
3. Por cada aserción $C(a)$ en $\mathcal{K}$ ponemos $\mathcal{L}(a) \leftarrow C$.
4. Por cada aserción $R(a, b)$ en $\mathcal{K}$ ponemos $\mathcal{L}(a, b) \leftarrow R$.

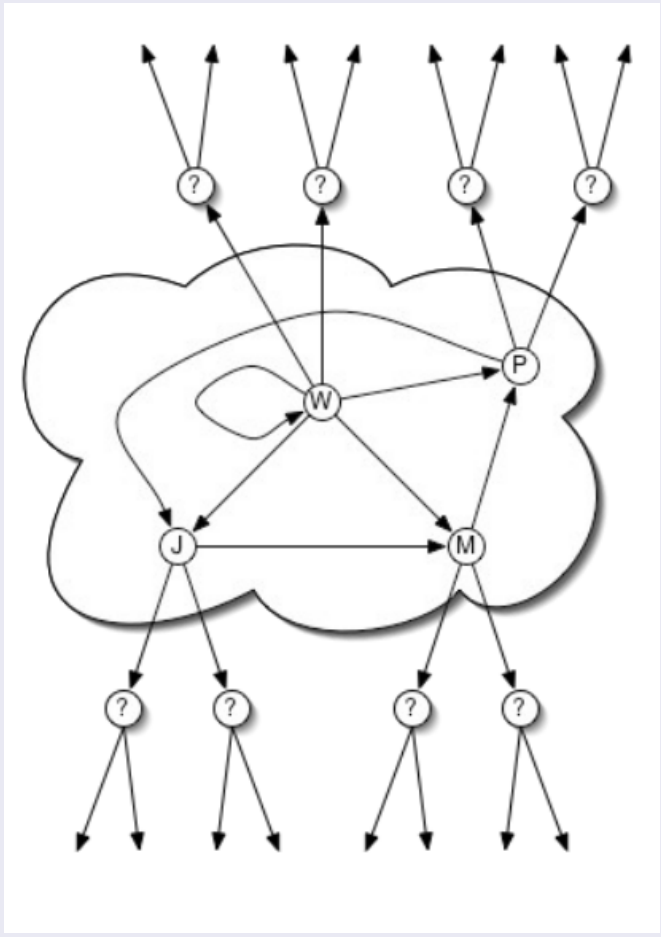
Ejemplo de inicialización

Sea una base $\mathcal{K}$ de conocimiento con el axioma $\neg A \sqcup \forall S.B$ y las aserciones $A(a)$, $\exists R.B(a)$, $R(a,b)$. $R(a,c)$, $S(b,b)$, $(A \sqcup B)(c)$.



Grafo cuando tenemos Abox

- Los individuos de la ontologia forman un grafo arbitrario
- Variables son nodos de árboles cuya raiz son individuos de la ontologia



Chequeo de consistencia de $(\mathcal{T}, \mathcal{A})$ cuando $\mathcal{T}$ es primitiva

1. Transformar a FNN la Abox $\mathcal{A}$,

$$\mathsf{FNN}(\mathcal{A}) = \bigcup_{C(a)\in\mathcal{A}} \mathsf{FNN}(C)(a) \cup \bigcup_{R(a,b)\in\mathcal{A}} R(a,b)$$

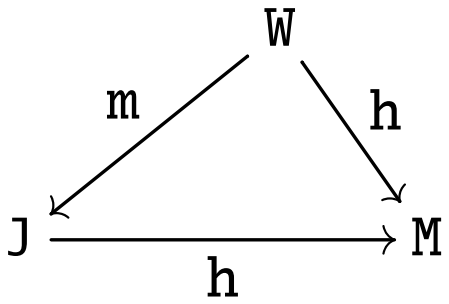
2. Transformar a FNN la Tbox primitiva

$$\mathsf{FNN}(\mathcal{T}) = \bigcup_{C\sqsubseteq D\in\mathcal{T}} \mathsf{FNN}(C) \sqsubseteq \mathsf{FNN}(D)$$

3. Inizializar el grafo a partir de la Abox $\mathsf{FNN}(\mathcal{A})$.
4. Aplicar las reglas de expansión para $\mathcal{ALC}$ con lazy unfolding usando $\mathsf{FNN}(\mathcal{T})$.

Ejemplo de una Tbox primitiva junto con Abox

$\mathcal{T}$	HappyParent =	Person⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)
$\mathcal{A}$	HappyParent(John) hasChild(John, Mary) marriedTo(Wendy, John)	¬Doctor(Mary) hasChild(Wendy, Mary)

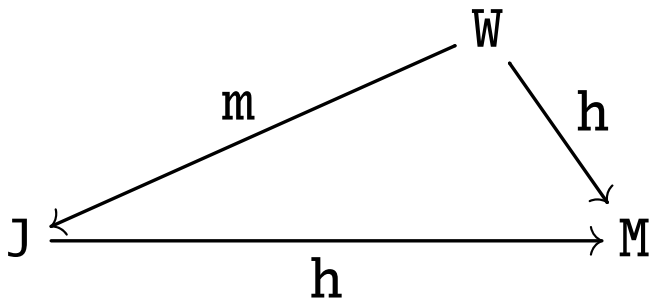


$\mathcal{L}(\mathtt{M}) = \{\neg \mathtt{D}\}$

$\mathcal{L}(\mathtt{J}) = \{\mathtt{H}\}$

Ejemplo de una Tbox primitiva junto con Abox

$\mathcal{T}$	HappyParent =	Person⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)
$\mathcal{A}$	HappyParent(John) hasChild(John, Mary) marriedTo(Wendy, John)	¬Doctor(Mary) hasChild(Wendy, Mary)

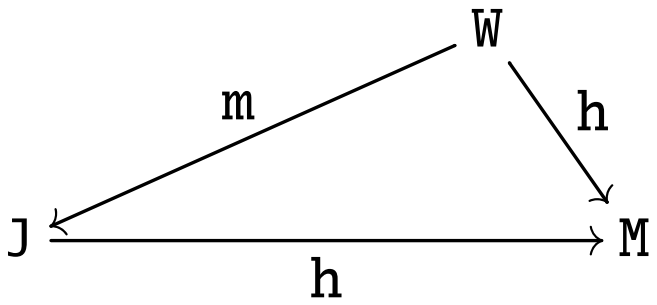


$\mathcal{L}(\mathtt{M}) = \{\neg \mathtt{D}\}$

$\mathcal{L}(\mathtt{J}) = \{\mathtt{H}, \mathtt{P}, \forall \mathtt{h}.(\mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D})\}$

Ejemplo de una Tbox primitiva junto con Abox

$\mathcal{T}$	HappyParent =	Person⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)
$\mathcal{A}$	HappyParent(John) hasChild(John, Mary) marriedTo(Wendy, John)	¬Doctor(Mary) hasChild(Wendy, Mary)

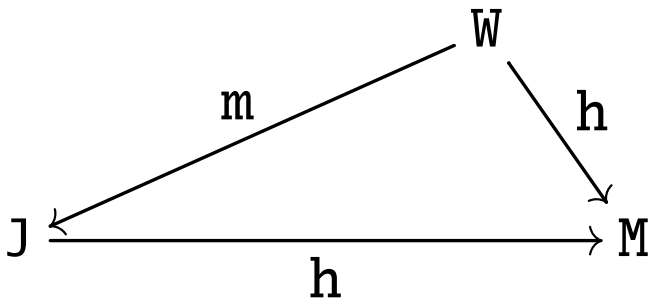


$$\mathcal{L}(\mathtt{M}) = \{\neg \mathtt{D}, \mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D}\}$$

$$\mathcal{L}(\mathtt{J}) = \{\mathtt{H}, \mathtt{P}, \forall \mathtt{h}.(\mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D})\}$$

Ejemplo de una Tbox primitiva junto con Abox

$\mathcal{T}$	HappyParent =	Person⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)
$\mathcal{A}$	HappyParent(John) hasChild(John, Mary) marriedTo(Wendy, John)	¬Doctor(Mary) hasChild(Wendy, Mary)

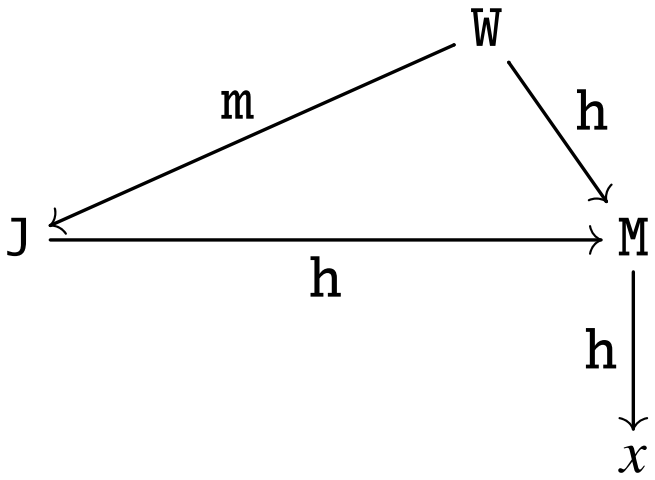


$\mathcal{L}(\mathtt{M}) = \{\neg \mathtt{D}, \mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D}, \exists \mathtt{h}.\mathtt{D}\}$

$\mathcal{L}(\mathtt{J}) = \{\mathtt{H}, \mathtt{P}, \forall \mathtt{h}.(\mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D})\}$

Ejemplo de una Tbox primitiva junto con Abox

$\mathcal{T}$	HappyParent =	Person⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)
$\mathcal{A}$	HappyParent(John) hasChild(John, Mary) marriedTo(Wendy, John)	¬Doctor(Mary) hasChild(Wendy, Mary)



$$\mathcal{L}(\mathtt{M}) = \{\neg \mathtt{D}, \mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D}, \exists \mathtt{h}.\mathtt{D}\}$$

$$\mathcal{L}(x) = \{\mathtt{D}\}$$

$$\mathcal{L}(\mathtt{J}) = \{\mathtt{H}, \mathtt{P}, \forall \mathtt{h}.(\mathtt{D} \sqcup \exists \mathtt{h}.\mathtt{D})\}$$

Ejemplo de Tbox general donde lazy unfolding no funciona

Sea la base de conocimiento:

$$A \sqcup B \sqsubseteq \forall R.\neg H \quad \neg A \sqsubseteq B \quad H(b) \quad R(a,b)$$

La Tbox no es primitiva.

Ejemplo de Tbox general donde lazy unfolding no funciona

Sea la base de conocimiento:

$$A \sqcup B \sqsubseteq \forall R.\neg H \quad \neg A \sqsubseteq B \quad H(b) \quad R(a,b)$$

La Tbox no es primitiva. Aunque la reescribamos

$$\begin{array}{ll} X \equiv A \sqcup B & Y \equiv \neg A \\ X \sqsubseteq \forall R.\neg H & Y \sqsubseteq B \end{array} \quad H(b) \quad R(a,b)$$

no queda primitiva, e.g. para poder escribirla en Protégé.

Ejemplo de Tbox general donde lazy unfolding no funciona

Sea la base de conocimiento:

$$A \sqcup B \sqsubseteq \forall R.\neg H \quad \neg A \sqsubseteq B \quad H(b) \quad R(a,b)$$

La Tbox no es primitiva. Aunque la reescribamos

$$\begin{array}{ll} X \equiv A \sqcup B & Y \equiv \neg A \\ X \sqsubseteq \forall R.\neg H & Y \sqsubseteq B \end{array} \quad H(b) \quad R(a,b)$$

no queda primitiva, e.g. para poder escribirla en Protégé.

Las reglas del lazy unfolding no detectan la inconsistencia.

Ejemplo de Tbox general donde lazy unfolding no funciona

Sea la base de conocimiento:

$$A \sqcup B \sqsubseteq \forall R.\neg H \quad \neg A \sqsubseteq B \quad H(b) \quad R(a,b)$$

La Tbox no es primitiva. Aunque la reescribamos

$$\begin{array}{ll} X \equiv A \sqcup B & Y \equiv \neg A \\ X \sqsubseteq \forall R.\neg H & Y \sqsubseteq B \end{array} \quad H(b) \quad R(a,b)$$

no queda primitiva, e.g. para poder escribirla en Protégé.

Las reglas del lazy unfolding no detectan la inconsistencia.

Para Tboxes generales, necesitamos otra regla de expansión mas general que el lazy unfolding.

Tbox General

Conversión a FNN de una Tbox general

$$\mathsf{FNN}(\mathcal{T}) \quad = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{FNN}(\neg C \sqcup D)$$

Regla de expansión para una Tbox general

Regla $\mathcal{T}$:
Si $C \in \mathcal{T}$ y $C \notin \mathcal{L}(x)$ entonces agregar C a $\mathcal{L}(x)$.

Reglas de expansión del algoritmo de tableaux para $\mathcal{ALC}$

Después de la inicialización, el algoritmo procede a aplicar las reglas de expansión en forma no determinística:

Regla $\sqcap$: Si $C \sqcap D \in \mathcal{L}(x)$ y $\{C, D\} \nsubseteq \mathcal{L}(x)$ entonces ponemos $\mathcal{L}(x) \leftarrow \{C, D\}$.

Regla $\sqcup$: Si $C \sqcup D \in \mathcal{L}(x)$ y $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ entonces ponemos $\mathcal{L}(x) \leftarrow \{C\}$ o $\mathcal{L}(x) \leftarrow \{D\}$.

Regla $\exists$: Si $\exists R.C \in \mathcal{L}(x)$ y
no existe y tal que $R \in \mathcal{L}(x, y)$ y $C \in \mathcal{L}(y)$ entonces
1. Agregamos un nodo con etiqueta y
(donde y es un nombre nuevo),
2. ponemos $\mathcal{L}(x, y) = \{R\}$,
3. ponemos $\mathcal{L}(y) = \{C\}$.

Regla $\forall$: Si $\forall R.C \in \mathcal{L}(x)$ y
existe y tal que $R \in \mathcal{L}(x, y)$ y $C \notin \mathcal{L}(y)$ entonces
ponemos $\mathcal{L}(y) \leftarrow C$

Regla $\mathcal{T}$: Si $C \in \mathcal{T}$ y $C \notin \mathcal{L}(x)$ entonces $\mathcal{L}(x) \leftarrow C$.

Ejemplo

Sea la base de conocimiento:

$$A \sqcup B \sqsubseteq \forall R.\neg H \quad \neg A \sqsubseteq B \quad H(b) \quad R(a,b)$$

1. Convertimos la base a FNN.

$$\neg A \sqcap \neg B \sqcup \forall R.\neg H \quad A \sqcup B \quad H(b) \quad R(a,b)$$

2. Inicializamos el grafo.
3. Aplicamos las reglas de expansión hasta llegar a un grafo completo sin contradicciones o hasta que todas las opciones nos den una contradicción.

Otros Ejemplos

El algoritmo dice si existe un modelo o no.

1. Sea $\mathcal{K}$ la base de conocimiento: $C(a)$, $\neg C \sqcap D(a)$. Queremos saber si es consistente.
2. Sea $\mathcal{K}$ la base de conocimiento: $\neg C \sqcap D$, $C(a)$,$\neg D(a)$. Queremos saber si es consistente.
3. Sea $\mathcal{K}$ la base de conocimiento: $C(a)$, $C \sqsubseteq \exists R.D$, $D \sqsubseteq E$. Queremos saber si $\mathcal{K} \models (\exists R.E)(a)$.

Hacer en el pizzarrón.

Otro Ejemplo

Sea $\mathcal{K} = (\emptyset, \mathcal{A})$ donde

$$\begin{aligned}\mathcal{A} = \quad & \{\texttt{tieneHijo}(\texttt{mario}, \texttt{luis}),\\ & \texttt{tieneHijo}(\texttt{mario}, \texttt{jorge}),\\ & \texttt{Hombre}(\texttt{luis}),\\ & \texttt{Hombre}(\texttt{jorge})\}\end{aligned}$$

Mostrar que $\mathcal{K} \not\models \forall \texttt{tieneHijo}.\texttt{Hombre}(\texttt{mario})$.

Este es un ejemplo de mundo abierto.

Ejemplo para el cual el algoritmo no termina

Sea $\mathcal{K}$ la base de conocimiento con el axioma $\exists R.\top$ en la TBox y la aserción $\top(a)$ en la ABox.

$$a \xrightarrow[R]{} x_1 \xrightarrow[R]{} x_2 \xrightarrow[R]{} \dots$$

$$\mathcal{L}(a) = \{\top, \exists R.\top\} \qquad \mathcal{L}(x_1) = \{\top, \exists R.\top\} \qquad \mathcal{L}(x_2) = \{\top, \exists R.\top\}$$

Modelo infinito.

- $\Delta^{\mathcal{I}} = \{a\} \cup \{x_i \mid i \in \mathbb{N}\}$
- $R^{\mathcal{I}} = \{(a, x_1)\} \cup \{(x_i, x_{i+1}) \mid i \in \mathbb{N}\}$.

Ejemplo para el cual el algoritmo no termina

$\mathtt{Person} = \exists \mathtt{hasParent}.\mathtt{Person} \quad \mathtt{Person}(\mathtt{John})$

$$\mathtt{John} \xrightarrow[\mathtt{hasParent}]{} x_1 \xrightarrow[\mathtt{hasParent}]{} x_2 \xrightarrow[\mathtt{hasParent}]{} \dots$$

$$\begin{aligned}
\mathcal{L}(\mathtt{John}) &= \{\mathtt{Person}, \exists \mathtt{hasParent}.\mathtt{Person}\} \\
\mathcal{L}(x_1) &= \{\mathtt{Person}, \exists \mathtt{hasParent}.\mathtt{Person}\} \\
\mathcal{L}(x_2) &= \{\mathtt{Person}, \exists \mathtt{hasParent}.\mathtt{Person}\} \\
&\vdots
\end{aligned}$$

Bloqueo de nodos

Predecesor

x es un predecesor de y si $\mathcal{L}(x, y) \neq \emptyset$.

Ancestro

x es un ancestro de y si:

1. x es un predecesor de y,
2. x es un predecesor de z y z es un ancestro de y.

Bloqueo

Un nodo cuya etiqueta es x se dice que está bloqueado por un nodo con etiqueta y si

1. x es una variable (no es un individuo),
2. y es un ancestro de x,
3. $\mathcal{L}(x) \subseteq \mathcal{L}(y)$.

Ejemplo donde se necesita el bloqueo

Sea $\mathcal{K}$ la base de conocimiento con el axioma $\exists R.\top$ en la TBox y la aserción $\top(a)$ en la ABox.

$$a \xrightarrow[R]{} x$$

$$\mathcal{L}(a) = \{\top, \exists R.\top\} \qquad \mathcal{L}(x) = \{\top\}$$

x está bloqueado por a ya que $\mathcal{L}(x) \subseteq \mathcal{L}(a)$.

Modelo finito.

- $\Delta^{\mathcal{I}} = \{a\}$
- $R^{\mathcal{I}} = \{(a, a)\}$.

Reglas de expansión CORREGIDAS para $\mathcal{ALC}$

Regla $\sqcap$: Si $C \sqcap D \in \mathcal{L}(x)$ y $\{C, D\} \not\subseteq \mathcal{L}(x)$ entonces
ponemos $\mathcal{L}(x) \leftarrow \{C, D\}$.

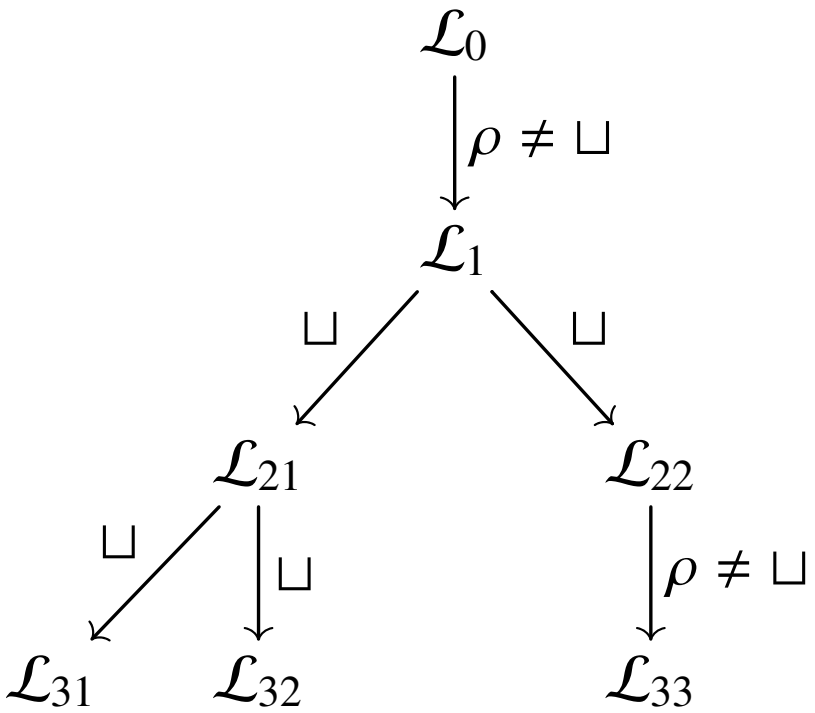
Regla $\sqcup$: Si $C \sqcup D \in \mathcal{L}(x)$ y $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ entonces
ponemos $\mathcal{L}(x) \leftarrow \{C\}$ o $\mathcal{L}(x) \leftarrow \{D\}$.

Regla $\exists$: Si x no está bloqueado, $\exists R.C \in \mathcal{L}(x)$ y
no existe y tal que $R \in \mathcal{L}(x, y)$ y $C \in \mathcal{L}(y)$ entonces
1. Agregamos un nodo con etiqueta y
(donde y es un nombre nuevo),
2. ponemos $\mathcal{L}(x, y) = \{R\}$,
3. ponemos $\mathcal{L}(y) = \{C\}$.

Regla $\forall$: Si $\forall R.C \in \mathcal{L}(x)$ y
existe y tal que $R \in \mathcal{L}(x, y)$ y $C \notin \mathcal{L}(y)$ entonces
ponemos $\mathcal{L}(x) \leftarrow C$

Regla $\mathcal{T}$: Si $C \in \mathcal{T}$ y $C \notin \mathcal{L}(x)$ entonces $\mathcal{L}(x) \leftarrow C$.

Transformación del grafo



Si llegamos a $\mathcal{L}_{31}$, $\mathcal{L}_{32}$, $\mathcal{L}_{33}$ donde todas tienen contradicciones, entonces la base original es inconsistente.

Notación. Sea $\rho \in \{\sqcap, \sqcup, \exists, \forall, \mathcal{T}\}$ una de las reglas del algoritmo. $\mathcal{L} \rightarrow_\rho \mathcal{L}'$ si $\mathcal{L}'$ se obtiene de $\mathcal{L}$ luego de aplicar la regla ρ.

¿ Cuando paramos de aplicar las reglas?

- $\mathcal{L}$ tiene una contradicción cuando existen a y A tal que $\{A, \neg A\} \subseteq \mathcal{L}(a)$.
- $(\mathcal{T}, \mathcal{L})$ está completo cuando no se le pueden aplicar mas reglas.

Condición de terminación del algoritmo

El algoritmo termina cuando se llega a

- a un grafo $\mathcal{L}$ completo y sin contradicciones. En este caso, la base de conocimientos $\mathcal{K}$ es consistente.
- cuando todas las posibilidades nos dan un grafo con contradicciones. En este caso, la base de conocimientos $\mathcal{K}$ es inconsistente.

Ejemplos donde se necesita el bloqueo

1. Sea $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.
   $\mathcal{T} = \{\texttt{Humano} \sqsubseteq \exists \texttt{tienePadre}.\texttt{Humano}\}$,
   $\mathcal{A} = \{\texttt{Ave}(\texttt{picaflor})\}$

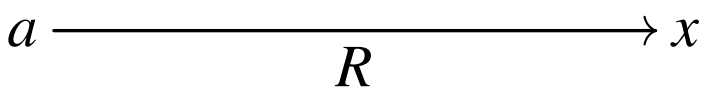
   $\mathcal{K} \models \neg\texttt{Humano}(\texttt{picaflor})?$

2. Sea $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
   $\mathcal{T} = \{C \sqsubseteq \forall S.A, A \sqsubseteq \exists R.\exists S.A, A \sqsubseteq \exists R.C\}$
   $\mathcal{A} = \{C(a), C(c), R(a,b), R(a,c), S(a,a), S(c,b)\}$

   $\mathcal{K} \models \exists R.\exists R.\exists S.A(a)?$

Ejemplo: el bloqueo se puede romper

Sea una base $\mathcal{K}$ de conocimiento con el axioma $A \sqsubseteq \exists R.A$ y las aserciones $A(a)$, $\forall R.\forall R.\neg A(a)$.



$$\mathcal{L}(a) = \{A, \forall R.\exists R.\neg A, \exists R.A\} \qquad \mathcal{L}(x) = \{A, \exists R.A\}$$

x está bloqueado, $\mathcal{L}(x) \subseteq \mathcal{L}(a)$, antes de aplicar la regla $\forall$.

Ejemplo: el bloqueo se puede romper

Sea una base $\mathcal{K}$ de conocimiento con el axioma $A \sqsubseteq \exists R.A$ y las aserciones $A(a)$, $\forall R.\forall R.\neg A(a)$.

$$a \xrightarrow[R]{} x$$

$$\mathcal{L}(a) = \{A, \forall R.\exists R.\neg A, \exists R.A\} \qquad \mathcal{L}(x) = \{A, \exists R.A\}$$

x está bloqueado, $\mathcal{L}(x) \subseteq \mathcal{L}(a)$, antes de aplicar la regla $\forall$.

$$a \xrightarrow[R]{} x$$

$$\mathcal{L}(a) = \{A, \forall R.\exists R.\neg A, \exists R.A\} \qquad \mathcal{L}(x) = \{A, \exists R.A, \exists R.\neg A\}$$

Ejemplo: el bloqueo se puede romper

Sea una base $\mathcal{K}$ de conocimiento con el axioma $A \sqsubseteq \exists R.A$ y las aserciones $A(a)$, $\forall R.\forall R.\neg A(a)$.

$$\mathcal{L}(a) = \{A, \forall R.\forall R.\neg A, \exists R.A\}$$

$$\mathcal{L}(x) = \{A, \exists R.A, \forall R.\neg A\}$$

$$\mathcal{L}(y) = \{A, \neg A\}$$

Podemos poner un orden en el que se evaluan las reglas (pero no es necesario para $\mathcal{ALC}$): reglas no generadoras de nodos se podrían aplicar antes de las generadoras.

Subconceptos

Subconcepto. Concepto que forma parte de la sintaxis de otro concepto. Por ejemplo `FiguraReligiosa` es un subconcepto de

$$\texttt{Pintura} \sqcap \exists \texttt{tieneFigura}.\texttt{FiguraReligiosa}$$

m cardinalidad del conjunto de subconceptos de todos los conceptos en $\mathcal{K}$

$m \leq n$ donde n es el tamaño de $\mathcal{K}$

Terminación

El algoritmo termina porque el grafo

1. el grado máximo de salida es m
2. el largo del camino máximo es 2^m

Observaciones:

1. *Las reglas de expansión nunca borran conceptos:* para todo x definido en $\mathcal{L}$ se tiene que $\mathcal{L}(x) \subseteq \mathcal{L}'(x)$ si $\mathcal{L} \rightarrow_\rho \mathcal{L}'$.
2. $C \in \mathcal{L}(x)$ entonces C es un subconcepto de algún concepto de $\mathcal{K}$.
3. Hay a lo sumo 2^m conjuntos diferentes de la forma $\mathcal{L}(x)$
4. Todas las reglas agregan conceptos. Pero solamente la regla del $\exists$ genera nuevos nodos.

m la cardinalidad del conjunto de subconceptos de todos los conceptos en $\mathcal{K}$

Adecuación

Adecuación

Si $(\mathcal{T}, \mathcal{A})$ es consistente entonces el algoritmo termina dando un grafo completo y sin contradicciones.

Idea de Demostración

Parecido a la demostración de adecuación del algoritmo de tableau para la satisfactibilidad de un concepto.
Las reglas de expansión preservan el "tener modelo".
El modelo en realidad no cambia excepto por la regla del existencial que lo extendemos con el nuevo individuos.

Completitud

Completitud

Si el algoritmo con entrada $(\mathcal{T}, \mathcal{A})$ termina dando un grafo completo y sin contradicciones entonces $(\mathcal{T}, \mathcal{A})$ es consistente.

Idea de la demostración

$\mathcal{L}$ grafo completo sin contradicciones
La interpretación $\mathcal{I}_c$ inducida por $\mathcal{L}$ se define:

- $\Delta^{\mathcal{I}_c} = \{x \mid x \text{ es un nodo no bloqueado de } \mathcal{L}\}$
- $A^{\mathcal{I}_c} = \{x \in \Delta^{\mathcal{I}_c} \mid A \in \mathcal{L}(x)\}$
- $R^{\mathcal{I}_c} = \{(x, y) \in \Delta^{\mathcal{I}_c} \times \Delta^{\mathcal{I}_c} \mid R \in \mathcal{L}(x, y)\} \cup$
  $\{(x, z) \in \Delta^{\mathcal{I}_c} \times \Delta^{\mathcal{I}_c} \mid R \in \mathcal{L}(x, y)\ y \text{ está bloqueado por } z\}$
- $a^{\mathcal{I}_c} = a.$

Complejidad del algoritmo de tableau

Complejidad es doble exponencial

- la profundidad de $\mathcal{L}$ esta acotada por 2^m
- el grado de salida esta acotado por m

Bibliografía para esta clase

- Franz Baader, Martin Bucheit, Bernard Hollunder. Cardinality Restrictions on Concepts. Artif. Intell. 88(1-2): 195-213 (1996)

- Capítulo 2 *Description Logic Handbook: Theory, Implementation, and Applications.* Editores Franz Baader and Diego Calvanese and Deborah L. McGuinness and Daniele Nardi and Peter F. Patel-Schneider. 2003.

- Franz Baader, Ian Horrocks, Ulrike Sattler: Description Logics. Handbook of Knowledge Representation.