Lógica Computacional y Demostración Automática

Carlos Areces

areces@loria.fr

http://www.loria.fr/~areces

INRIA Nancy Grand Est, France

Diciembre 2008

Lógicas para la Descripción

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.
- Descienden originalmente del trabajo en redes semánticas (semantic networks) de Quillian y el paradigma de Frames de Minsky.

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.
- Descienden originalmente del trabajo en redes semánticas (semantic networks) de Quillian y el paradigma de Frames de Minsky.
- Las principales características de estos lenguajes son:
  - Un lenguaje simple de usar (una extensión del lenguaje proposicional, sin variables de estado);

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.
- Descienden originalmente del trabajo en redes semánticas (semantic networks) de Quillian y el paradigma de Frames de Minsky.
- Las principales características de estos lenguajes son:
  - Un lenguaje simple de usar (una extensión del lenguaje proposicional, sin variables de estado);
  - Que incluye una noción restringida de cuantificación;

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.
- Descienden originalmente del trabajo en redes semánticas (semantic networks) de Quillian y el paradigma de Frames de Minsky.
- Las principales características de estos lenguajes son:
  - Un lenguaje simple de usar (una extensión del lenguaje proposicional, sin variables de estado);
  - Que incluye una noción restringida de cuantificación;
  - Con operadores especialmente elegidos para facilitar la creación de definiciones;

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.
- Descienden originalmente del trabajo en redes semánticas (semantic networks) de Quillian y el paradigma de Frames de Minsky.
- Las principales características de estos lenguajes son:
  - Un lenguaje simple de usar (una extensión del lenguaje proposicional, sin variables de estado);
  - Que incluye una noción restringida de cuantificación;
  - Con operadores especialmente elegidos para facilitar la creación de definiciones;
  - Con un buen balance entre expresividad y tratabilidad;

Lógicas para la Descripción

- Las lógicas para la descripción (description logics, DL) son lenguajes formales especialmente diseñados para la representación del conocimiento.
- Descienden originalmente del trabajo en redes semánticas (semantic networks) de Quillian y el paradigma de Frames de Minsky.
- Las principales características de estos lenguajes son:
  - Un lenguaje simple de usar (una extensión del lenguaje proposicional, sin variables de estado);
  - Que incluye una noción restringida de cuantificación;
  - Con operadores especialmente elegidos para facilitar la creación de definiciones;
  - Con un buen balance entre expresividad y tratabilidad;
  - Que cuenta con sistemas de inferencia altamente optimizados.

Lógicas para la Descripción

Lógicas para la Descripción

En una DL tenemos operadores para construir definiciones usando conceptos, roles e individuos:

- Los conceptos corresponden a subconjuntos del universo.

Lógicas para la Descripción

En una DL tenemos operadores para construir definiciones usando conceptos, roles e individuos:

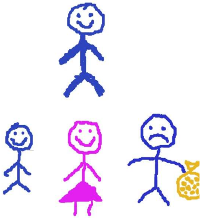
- Los conceptos corresponden a subconjuntos del universo.
- Los roles corresponden a relaciones binarias en el universo.

Lógicas para la Descripción

En una DL tenemos operadores para construir definiciones usando conceptos, roles e individuos:

- Los conceptos corresponden a subconjuntos del universo.
- Los roles corresponden a relaciones binarias en el universo.
- Los individuos corresponden to elementos del universo.

Ejemplo: El “Padre Contento”



$$\text{Conceptos} = \{\ \mathsf{Hombre},\ \mathsf{Mujer},\ \mathsf{Contento},\ \mathsf{Rico}\ \}$$

$$\text{Roles} = \{\ \mathsf{tiene\text{-}hijo}\ \}$$

$$\begin{aligned}\mathsf{PadreContento} \equiv\ & \mathsf{Hombre} \sqcap \\ & \exists\, \mathsf{tiene\text{-}hijo.Hombre} \sqcap \\ & \exists\, \mathsf{tiene\text{-}hijo.Mujer} \sqcap \\ & \forall\, \mathsf{tiene\text{-}hijo.(Contento} \sqcup \mathsf{Rico)}\end{aligned}$$

$\mathsf{carlos}{:}\neg\mathsf{PadreContento}$

Areas de Aplicación

Areas de Aplicación

- Bases de conocimiento terminológicas y ontologías
  - DLs fueron desarrolladas para esta tarea
  - Especialmente útiles como lenguaje de definición y mantenimiento de ontologías

Areas de Aplicación

- Bases de conocimiento terminológicas y ontologías
  - DLs fueron desarrolladas para esta tarea
  - Especialmente útiles como lenguaje de definición y mantenimiento de ontologías
- Applicaciones en Bases de Datos
  - DLs pueden capturar la semantica de varias metodologías de modelado en BD e.g., diagramas de ER, UML, etc
  - los motores de inferencia DL pueden usarse para proveer soporte durante el diseño de diagramas, mantenimiento y consulta

Areas de Aplicación

Areas de Aplicación

- Web Semántica
  - Agregar ‘markup semántico’ a la información en la web
  - Este markup usaría repositorios de ontologías como repositorío común de definiciones con una semántica clara
  - los motores de inferencia DL se usarían para el desarrollo, mantenimiento y fusión de estas ontologías y para la evalucion dynámica de recursos (e.g., búsqueda).

Areas de Aplicación

- Web Semántica
  - Agregar ‘markup semántico’ a la información en la web
  - Este markup usaría repositorios de ontologías como repositorío común de definiciones con una semántica clara
  - los motores de inferencia DL se usarían para el desarrollo, mantenimiento y fusión de estas ontologías y para la evalucion dynámica de recursos (e.g., búsqueda).
- Linguística Computacional
  - Muchas tareas en linguística computacional requieren inferencia y ’background knowledge’: desde tareas puntuales como resolución de referencias a problemas generales como question-answering.
  - En ciertos casos, el poder expresivo ofrecido por DLs es suficiene y no necesitamos recurrir a LPO.

Lógicas para la Descripción

Lógicas para la Descripción

- El lenguaje se define en tres niveles.

Lógicas para la Descripción

- El lenguaje se define en tres niveles.
  - Conceptos: Por un lado construimos conceptos complejos usando conceptos (atómicos o creados via definición) y roles: E.g., $\exists$tiene-hijo.Hombre

Lógicas para la Descripción

- El lenguaje se define en tres niveles.
  - Conceptos: Por un lado construimos conceptos complejos usando conceptos (atómicos o creados via definición) y roles: E.g., $\exists$tiene-hijo.Hombre
  - Definiciones: Usamos conceptos para armar definiciones (o relaciones entre definiciones): E.g., PadreContento $\equiv \ldots$

Lógicas para la Descripción

- El lenguaje se define en tres niveles.
  - Conceptos: Por un lado construimos conceptos complejos usando conceptos (atómicos o creados via definición) y roles: E.g., $\exists$tiene-hijo.Hombre
  - Definiciones: Usamos conceptos para armar definiciones (o relaciones entre definiciones): E.g., PadreContento $\equiv \ldots$
  - Aserciones: Podemos asignar conceptos o roles a elementos particulares de nuestro modelo: E.g., carlos:$\neg$PadreContento

Lógicas para la Descripción

- El lenguaje se define en tres niveles.
  - Conceptos: Por un lado construimos conceptos complejos usando conceptos (atómicos o creados via definición) y roles: E.g., $\exists$tiene-hijo.Hombre
  - Definiciones: Usamos conceptos para armar definiciones (o relaciones entre definiciones): E.g., PadreContento $\equiv \ldots$
  - Aserciones: Podemos asignar conceptos o roles a elementos particulares de nuestro modelo: E.g., carlos:$\neg$PadreContento
- Una base de conocimiento es un par $\langle D, A\rangle$ con D un conjunto de definiciones (la "T-Box") y A un conjunto de aserciones (la "A-Box").

$\mathcal{ALC}$: Construcción de Conceptos

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.
  - Un concepto atómico: Hombre, Mujer

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.
  - Un concepto atómico: Hombre, Mujer
  - Operadores booleanos: Si C y D son conceptos entonces los siguientes son conceptos

$C \sqcap D$	la conjunción de C y D	Rico $\sqcap$ Apuesto

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.
  - Un concepto atómico: Hombre, Mujer
  - Operadores booleanos: Si C y D son conceptos entonces los siguientes son conceptos

$C \sqcap D$	la conjunción de C y D	Rico $\sqcap$ Apuesto
$C \sqcup D$	la disjunción de C y D	Rico $\sqcup$ Apuesto

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.
  - Un concepto atómico: Hombre, Mujer
  - Operadores booleanos: Si C y D son conceptos entonces los siguientes son conceptos

$C \sqcap D$	la conjunción de C y D	Rico $\sqcap$ Apuesto
$C \sqcup D$	la disjunción de C y D	Rico $\sqcup$ Apuesto
$\neg C$	la negación de C	$\neg$Politico

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.
  - Un concepto atómico: Hombre, Mujer
  - Operadores booleanos: Si C y D son conceptos entonces los siguientes son conceptos

$C \sqcap D$	la conjunción de C y D	Rico $\sqcap$ Apuesto
$C \sqcup D$	la disjunción de C y D	Rico $\sqcup$ Apuesto
$\neg C$	la negación de C	$\neg$Politico

  - Operadores relacionales: Si C es un concepto y R es un role, los siguientes son conceptos

$\forall R.C$	todo elem. acc. via R está en C	$\forall$hijo-de.Mujer

$\mathcal{ALC}$: Construcción de Conceptos

- Un concepto puede ser
  - $\top$, el concepto trivial, del que todo elemento es miembro.
  - Un concepto atómico: Hombre, Mujer
  - Operadores booleanos: Si C y D son conceptos entonces los siguientes son conceptos

$C \sqcap D$	la conjunción de C y D	Rico $\sqcap$ Apuesto
$C \sqcup D$	la disjunción de C y D	Rico $\sqcup$ Apuesto
$\neg C$	la negación de C	$\neg$Politico

  - Operadores relacionales: Si C es un concepto y R es un role, los siguientes son conceptos

$\forall R.C$	todo elem. acc. via R está en C	$\forall$hijo-de.Mujer
$\exists R.C$	un elem. acc. via R está en C	$\exists$hijo-de.Mujer

Construcción de Definiciones

Construcción de Definiciones

La T-Box es una lista de definiciones.

Construcción de Definiciones

La T-Box es una lista de definiciones.

Dados dos conceptos C y D, hay dos tipos de definiciones:

- Definiciones Parciales: $C \sqsubseteq D$. Las condiciones indicadas en C son suficientes para calificar a los elementos de C como miembros de D, pero no son condiciones necesarias; o vice-versa.

Construcción de Definiciones

La T-Box es una lista de definiciones.

Dados dos conceptos C y D, hay dos tipos de definiciones:

- Definiciones Parciales: $C \sqsubseteq D$. Las condiciones indicadas en C son suficientes para calificar a los elementos de C como miembros de D, pero no son condiciones necesarias; o vice-versa.

$\exists$hijo-de.Hombre $\sqcap$ $\exists$hijo-de.Mujer $\sqsubseteq$ PadreOcupado (condición suficiente)

Construcción de Definiciones

La T-Box es una lista de definiciones.

Dados dos conceptos C y D, hay dos tipos de definiciones:

- Definiciones Parciales: $C \sqsubseteq D$. Las condiciones indicadas en C son suficientes para calificar a los elementos de C como miembros de D, pero no son condiciones necesarias; o vice-versa.

$$\begin{array}{rcll} \exists \text{hijo-de}.\text{Hombre} \sqcap \exists \text{hijo-de}.\text{Mujer} & \sqsubseteq & \text{PadreOcupado} & \text{(condición suficiente)} \\ \text{PadreOcupado} & \sqsubseteq & \exists \text{hijo-de}.\top & \text{(condición necesaria)} \end{array}$$

Construcción de Definiciones

La T-Box es una lista de definiciones.

Dados dos conceptos C y D, hay dos tipos de definiciones:

- Definiciones Parciales: $C \sqsubseteq D$. Las condiciones indicadas en C son suficientes para calificar a los elementos de C como miembros de D, pero no son condiciones necesarias; o vice-versa.

$$\begin{array}{rcll} \exists \text{hijo-de}.\text{Hombre} \sqcap \exists \text{hijo-de}.\text{Mujer} & \sqsubseteq & \text{PadreOcupado} & \text{(condición suficiente)} \\ \text{PadreOcupado} & \sqsubseteq & \exists \text{hijo-de}.\top & \text{(condición necesaria)} \end{array}$$

- Definiciones Totales: $C \equiv D$. Las condiciones indicadas en D son necesarias y suficientes para calificar a los elementos de D como miembros de C. Los conceptos C y D son equivalentes.

Construcción de Definiciones

La T-Box es una lista de definiciones.

Dados dos conceptos C y D, hay dos tipos de definiciones:

- Definiciones Parciales: $C \sqsubseteq D$. Las condiciones indicadas en C son suficientes para calificar a los elementos de C como miembros de D, pero no son condiciones necesarias; o vice-versa.

$$\begin{array}{rcll} \exists\text{hijo-de}.\text{Hombre} \sqcap \exists\text{hijo-de}.\text{Mujer} & \sqsubseteq & \text{PadreOcupado} & \text{(condición suficiente)} \\ \text{PadreOcupado} & \sqsubseteq & \exists\text{hijo-de}.\top & \text{(condición necesaria)} \end{array}$$

- Definiciones Totales: $C \equiv D$. Las condiciones indicadas en D son necesarias y suficientes para calificar a los elementos de D como miembros de C. Los conceptos C y D son equivalentes.

$$\text{Abuela} \equiv \text{Mujer} \sqcap \exists\text{hijo-de}.\exists\text{hijo-de}.\top$$

Construcción de Aserciones

Construcción de Aserciones

Podemos “asignar propiedades” a elementos particulares de la situación que estamos describiendo en la A-Box.

Construcción de Aserciones

Podemos "asignar propiedades" a elementos particulares de la situación que estamos describiendo en la A-Box.

Dados elementos a y b, un concepto C y una relacion R

- Asignación de Elementos a Conceptos: $a{:}C$. Indica que el concepto C es applicable al elemento a. Es decir, todas las condiciones indicadas por C se aplican a a.

Construcción de Aserciones

Podemos “asignar propiedades” a elementos particulares de la situación que estamos describiendo en la A-Box.
Dados elementos a y b, un concepto C y una relacion R

- Asignación de Elementos a Conceptos: $a{:}C$. Indica que el concepto C es applicable al elemento a. Es decir, todas las condiciones indicadas por C se aplican a a.

carlos:Argentino

Construcción de Aserciones

Podemos “asignar propiedades” a elementos particulares de la situación que estamos describiendo en la A-Box.
Dados elementos a y b, un concepto C y una relacion R

- Asignación de Elementos a Conceptos: $a{:}C$. Indica que el concepto C es applicable al elemento a. Es decir, todas las condiciones indicadas por C se aplican a a.

$$\text{carlos}:\text{Argentino}$$

$$\text{carlos}:(\text{Argentino} \sqcap \exists\text{vive-en}.\text{Europa})$$

Construcción de Aserciones

Podemos "asignar propiedades" a elementos particulares de la situación que estamos describiendo en la A-Box.
Dados elementos a y b, un concepto C y una relacion R

- Asignación de Elementos a Conceptos: $a{:}C$. Indica que el concepto C es applicable al elemento a. Es decir, todas las condiciones indicadas por C se aplican a a.

$$\text{carlos}:\text{Argentino}$$

$$\text{carlos}:(\text{Argentino} \sqcap \exists\text{vive-en}.\text{Europa})$$

- Asignación de Relaciones entre Elementos: $(a, b){:}R$. Indica que los elementos a y b estan relacionados via el rol R.

Construcción de Aserciones

Podemos "asignar propiedades" a elementos particulares de la situación que estamos describiendo en la A-Box.
Dados elementos a y b, un concepto C y una relacion R

- Asignación de Elementos a Conceptos: $a{:}C$. Indica que el concepto C es applicable al elemento a. Es decir, todas las condiciones indicadas por C se aplican a a.

$$\text{carlos}{:}\text{Argentino}$$

$$\text{carlos}{:}(\text{Argentino} \sqcap \exists\text{vive-en}.\text{Europa})$$

- Asignación de Relaciones entre Elementos: $(a, b){:}R$. Indica que los elementos a y b estan relacionados via el rol R.

$$(\text{carlos},\text{nancy}){:}\text{vive-en}$$

Semántica: Modelos

Semántica: Modelos

Sea CON un conjunto de conceptos atómicos, ROL un conjunto de roles y IND un conjunto de individuos.

Semántica: Modelos

Sea CON un conjunto de conceptos atómicos, ROL un conjunto de roles y IND un conjunto de individuos.
Una interpretación o un modelo para $\mathcal{ALC}$ es un par $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ tal que

- $\Delta^{\mathcal{I}}$ es un conjunto no vacío arbitrario

Semántica: Modelos

Sea CON un conjunto de conceptos atómicos, ROL un conjunto de roles y IND un conjunto de individuos.
Una interpretación o un modelo para $\mathcal{ALC}$ es un par $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ tal que

- $\Delta^{\mathcal{I}}$ es un conjunto no vacío arbitrario
- $\cdot^{\mathcal{I}}$ es una función de interpretación de conceptos atómicos, roles e individuos tal que

$$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \text{ para } C \in \text{CON}$$

Semántica: Modelos

Sea CON un conjunto de conceptos atómicos, ROL un conjunto de roles y IND un conjunto de individuos.
Una interpretación o un modelo para $\mathcal{ALC}$ es un par $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ tal que

- $\Delta^{\mathcal{I}}$ es un conjunto no vacío arbitrario
- $\cdot^{\mathcal{I}}$ es una función de interpretación de conceptos atómicos, roles e individuos tal que

$$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \text{ para } C \in \text{CON}$$
$$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \text{ para } R \in \text{ROL}$$

Semántica: Modelos

Sea CON un conjunto de conceptos atómicos, ROL un conjunto de roles y IND un conjunto de individuos.

Una interpretación o un modelo para $\mathcal{ALC}$ es un par $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ tal que

- $\Delta^{\mathcal{I}}$ es un conjunto no vacío arbitrario
- $\cdot^{\mathcal{I}}$ es una función de interpretación de conceptos atómicos, roles e individuos tal que

$$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \text{ para } C \in \text{CON}$$
$$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \text{ para } R \in \text{ROL}$$
$$a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \text{ para } a \in \text{IND}$$

Semántica: Modelos

Sea CON un conjunto de conceptos atómicos, ROL un conjunto de roles y IND un conjunto de individuos.
Una interpretación o un modelo para $\mathcal{ALC}$ es un par $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ tal que

- $\Delta^{\mathcal{I}}$ es un conjunto no vacío arbitrario
- $\cdot^{\mathcal{I}}$ es una función de interpretación de conceptos atómicos, roles e individuos tal que

$$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \text{ para } C \in \text{CON}$$
$$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \text{ para } R \in \text{ROL}$$
$$a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \text{ para } a \in \text{IND}$$

(i.e., un modelo de DL no es otra cosa que un modelo de LPO para la signatura $\langle \text{CON} \cup \text{ROL}, \{\}, \text{IND} \rangle$)

Semántica: Conceptos

Semántica: Conceptos

Dado una interpretación $\mathcal{I}$ podemos definir la interpretación de un concepto arbitrario de $\mathcal{ALC}$ recursivamente como

$$
\begin{aligned}
(\neg C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}}
\end{aligned}
$$

Semántica: Conceptos

Dado una interpretación $\mathcal{I}$ podemos definir la interpretación de un concepto arbitrario de $\mathcal{ALC}$ recursivamente como

$$
\begin{array}{rcl}
(\neg C)^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & := & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} & := & \{ i \mid \exists j.(i,j) \in R^{\mathcal{I}} \text{ y } j \in C^{\mathcal{I}} \}
\end{array}
$$

Semántica: Conceptos

Dado una interpretación $\mathcal{I}$ podemos definir la interpretación de un concepto arbitrario de $\mathcal{ALC}$ recursivamente como

$$
\begin{array}{rcl}
(\neg C)^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & := & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} & := & \{ i \mid \exists j.(i,j) \in R^{\mathcal{I}} \text{ y } j \in C^{\mathcal{I}} \}
\end{array}
$$

$C \sqcup D$ es equivalente a $\neg(\neg C \sqcap \neg D)$ y
$(\forall R.C)$ es equivalente a $\neg(\exists R.\neg C)$.

Semántica: Definiciones y Aserciones

Dada una interpretación $\mathcal{I}$ decimos que

- $\mathcal{I}$ satisface una definición parcial $C \sqsubseteq D$ (definición total $C \equiv D$) sii

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad (C^{\mathcal{I}} = D^{\mathcal{I}})$$

Semántica: Definiciones y Aserciones

Dada una interpretación $\mathcal{I}$ decimos que

- $\mathcal{I}$ satisface una definición parcial $C \sqsubseteq D$ (definición total $C \equiv D$) sii

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad (C^{\mathcal{I}} = D^{\mathcal{I}})$$

- $\mathcal{I}$ satisface una T-Box T sii satisface todas las definiciones (parciales o totales) en T

Semántica: Definiciones y Aserciones

Dada una interpretación $\mathcal{I}$ decimos que

- $\mathcal{I}$ satisface una definición parcial $C \sqsubseteq D$ (definición total $C \equiv D$) sii
$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad (C^{\mathcal{I}} = D^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una T-Box T sii satisface todas las definiciones (parciales o totales) en T
- $\mathcal{I}$ satisface una asercion $a{:}C$ ($(a, b){:}R$) sii
$$a^{\mathcal{I}} \in C^{\mathcal{I}} \quad ((a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}})$$

Semántica: Definiciones y Aserciones

Dada una interpretación $\mathcal{I}$ decimos que

- $\mathcal{I}$ satisface una definición parcial $C \sqsubseteq D$ (definición total $C \equiv D$) sii
$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad (C^{\mathcal{I}} = D^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una T-Box T sii satisface todas las definiciones (parciales o totales) en T
- $\mathcal{I}$ satisface una assercion $a{:}C$ ($(a, b){:}R$) sii
$$a^{\mathcal{I}} \in C^{\mathcal{I}} \quad ((a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una A-Box A sii satisface todas las aserciones en A

Semántica: Definiciones y Aserciones

Dada una interpretación $\mathcal{I}$ decimos que

- $\mathcal{I}$ satisface una definición parcial $C \sqsubseteq D$ (definición total $C \equiv D$) sii
$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad (C^{\mathcal{I}} = D^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una T-Box T sii satisface todas las definiciones (parciales o totales) en T
- $\mathcal{I}$ satisface una asercion $a{:}C$ $((a,b){:}R)$ sii
$$a^{\mathcal{I}} \in C^{\mathcal{I}} \quad ((a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una A-Box A sii satisface todas las aserciones en A
- $\mathcal{I}$ satisface una KB $K = \langle T, A \rangle$ sii $\mathcal{I}$ satisface T y A.

Semántica: Definiciones y Aserciones

Dada una interpretación $\mathcal{I}$ decimos que

- $\mathcal{I}$ satisface una definición parcial $C \sqsubseteq D$ (definición total $C \equiv D$) sii
$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad (C^{\mathcal{I}} = D^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una T-Box T sii satisface todas las definiciones (parciales o totales) en T
- $\mathcal{I}$ satisface una assercion $a{:}C$ $((a, b){:}R)$ sii
$$a^{\mathcal{I}} \in C^{\mathcal{I}} \quad ((a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}})$$
- $\mathcal{I}$ satisface una A-Box A sii satisface todas las aserciones en A
- $\mathcal{I}$ satisface una KB $K = \langle T, A \rangle$ sii $\mathcal{I}$ satisface T y A.

Una KB K es consistente (o satisfacible) sii existe una interpretación $\mathcal{I}$ que la satisface.

Un Ejemplo Completo

Un Ejemplo Completo

$$\text{La T-Box:}\quad \begin{array}{rcl} \text{Mujer} & \sqsubseteq & \text{Persona} \sqcap \exists\text{sexo.Femenino} \\ \text{Hombre} & \sqsubseteq & \text{Persona} \sqcap \exists\text{sexo.Masculino} \\ \text{PadreOMadre} & \equiv & \text{Persona} \sqcap \exists\text{hijo-de.Persona} \\ \text{Madre} & \equiv & \text{Mujer} \sqcap \text{PadreOMadre} \\ \text{Padre} & \equiv & \text{Hombre} \sqcap \text{PadreOMadre} \end{array}$$

$$\text{La A-Box:}\quad \begin{array}{r} \text{alicia:Madre} \\ (\text{alicia,betty})\text{:hijo-de} \\ (\text{alicia,carlos})\text{:hijo-de} \end{array}$$

Sintaxis y Semántica de $\mathcal{ALC}$

Sintaxis y Semántica de $\mathcal{ALC}$

Constructor	Sintaxis	Semántica
concepto atómico	C	$C^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
negación ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunción	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disyunción	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
quant. universal ($\mathcal{U}$)	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \rightarrow d_2 \in C^{\mathcal{I}})\}$
quant. existencial ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}})\}$

Bases de Conocimiento

Bases de Conocimiento

Una base de conocimientos K es un par $K = \langle T, A \rangle$ tal que

- T es la T(erminological)-Box, un conjunto finito de expresiones de la forma $C \sqsubseteq D$. Las fórmulas en T se llaman terminological axioms.

Bases de Conocimiento

Una base de conocimientos K es un par $K = \langle T, A \rangle$ tal que

- T es la T(erminological)-Box, un conjunto finito de expresiones de la forma $C \sqsubseteq D$. Las fórmulas en T se llaman terminological axioms.
- A es la A(ssertional)-Box, un conjunto finito de expresiones de la forma $a{:}C$ o $(a, b){:}R$. Las fórmulas en A se llaman assertions.

Bases de Conocimiento

Una base de conocimientos K es un par $K = \langle T, A \rangle$ tal que

- T es la T(erminological)-Box, un conjunto finito de expresiones de la forma $C \sqsubseteq D$. Las fórmulas en T se llaman terminological axioms.
- A es la A(ssertional)-Box, un conjunto finito de expresiones de la forma $a{:}C$ o $(a, b){:}R$. Las fórmulas en A se llaman assertions.

Sea $\mathcal{I}$ una interpretación y φ un axioma terminológico o una asercion. Decimos que $\mathcal{I} \models \varphi$ si

- $\varphi = C \sqsubseteq D$ y $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, o
- $\varphi = a{:}C$ y $a^{\mathcal{I}} \in C^{\mathcal{I}}$, o
- $\varphi = (a, b){:}R$ y $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Decimos que $\langle T, A \rangle$ es satisfacible si existe una interpretación $\mathcal{I}$ que satisface T y A.

Tareas de Inferencia

Tareas de Inferencia

La tarea básica de inferencia es satisfiabilidad de conceptos:

INPUT: Un concepto C

OUTPUT: Si, si existe una interpretación $\mathcal{I}$ tal que $C^{\mathcal{I}} \neq \{\}$
No, en otro caso.

Tareas de Inferencia

La tarea básica de inferencia es satisfiabilidad de conceptos:

INPUT: Un concepto C

OUTPUT: Si, si existe una interpretación $\mathcal{I}$ tal que $C^{\mathcal{I}} \neq \{\}$
No, en otro caso.

Equivalentemente, subsumpción de conceptos:

INPUT: un par de conceptos (C, D)

OUTPUT: Si, si para toda interpretación $\mathcal{I}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
No, en otro caso.

Tareas de Inferencia

La tarea básica de inferencia es satisfiabilidad de conceptos:

INPUT:	Un concepto C
OUTPUT:	Si, si existe una interpretación $\mathcal{I}$ tal que $C^{\mathcal{I}} \neq \{\}$
	No, en otro caso.

Equivalentemente, subsumpción de conceptos:

INPUT:	un par de conceptos (C, D)
OUTPUT:	Si, si para toda interpretación $\mathcal{I}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
	No, en otro caso.

T: Satisfiabilidad y subsumpción son interdefinibles (en un lenguaje con conjunción y negación).

Tareas de Inferencia

La tarea básica de inferencia es satisfiabilidad de conceptos:

INPUT: Un concepto C

OUTPUT: Si, si existe una interpretación $\mathcal{I}$ tal que $C^{\mathcal{I}} \neq \{\}$
No, en otro caso.

Equivalentemente, subsumpción de conceptos:

INPUT: un par de conceptos (C, D)

OUTPUT: Si, si para toda interpretación $\mathcal{I}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
No, en otro caso.

T: Satisfiabilidad y subsumpción son interdefinibles (en un lenguaje con conjunción y negación).

D: $[\Rightarrow]$ C no es satisfacible sii $C \sqsubseteq \neg\top$

Tareas de Inferencia

La tarea básica de inferencia es satisfiabilidad de conceptos:

INPUT: Un concepto C

OUTPUT: Si, si existe una interpretación $\mathcal{I}$ tal que $C^{\mathcal{I}} \neq \{\}$
No, en otro caso.

Equivalentemente, subsumpción de conceptos:

INPUT: un par de conceptos (C, D)

OUTPUT: Si, si para toda interpretación $\mathcal{I}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
No, en otro caso.

T: Satisfiabilidad y subsumpción son interdefinibles (en un lenguaje con conjunción y negación).

D: [$\Rightarrow$] C no es satisfacible sii $C \sqsubseteq \neg\top$
[$\Leftarrow$] $C \sqsubseteq D$ sii $C \sqcap \neg D$ no es satisfacible

Tareas de Inferencia Respecto de una Base de Conocimiento

Tareas de Inferencia Respecto de una Base de Conocimiento

Sea T una base de conocimientos, $C_1, C_2 \in$ CON, $R \in$ ROL y $a, b \in$ IND, podemos definir las siguientes *tareas de inferencia*

- Subsumption, $T \models C_1 \sqsubseteq C_2$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.

Tareas de Inferencia Respecto de una Base de Conocimiento

Sea T una base de conocimientos, $C_1, C_2 \in$ CON, $R \in$ ROL y $a, b \in$ IND, podemos definir las siguientes *tareas de inferencia*

- Subsumption, $T \models C_1 \sqsubseteq C_2$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- Instance Checking, $T \models a{:}C$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $a^{\mathcal{I}} \in C^{\mathcal{I}}$.

Tareas de Inferencia Respecto de una Base de Conocimiento

Sea T una base de conocimientos, $C_1, C_2 \in \mathrm{CON}$, $R \in \mathrm{ROL}$ y $a, b \in \mathrm{IND}$, podemos definir las siguientes *tareas de inferencia*

- Subsumption, $T \models C_1 \sqsubseteq C_2$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- Instance Checking, $T \models a{:}C$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- Concept Consistency ($T \not\models C \doteq \bot$). Chequear si para alguna interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos qeu $C^{\mathcal{I}} \neq \{\}$.

Tareas de Inferencia Respecto de una Base de Conocimiento

Sea T una base de conocimientos, $C_1, C_2 \in$ CON, $R \in$ ROL y $a, b \in$ IND, podemos definir las siguientes *tareas de inferencia*

- Subsumption, $T \models C_1 \sqsubseteq C_2$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- Instance Checking, $T \models a{:}C$. Chequear si para toda interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos que $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- Concept Consistency ($T \not\models C \doteq \bot$). Chequear si para alguna interpretación $\mathcal{I}$ tal que $\mathcal{I} \models T$ tenemos qeu $C^{\mathcal{I}} \neq \{\}$.

Otras: Relation Checking ($T \models (a, b){:}R$), Knowledge Base Consistency ($T \not\models \bot$), etc.

Tareas de Inferencia Complejas

Tareas de Inferencia Complejas

Otras tareas de inferencia mas complejas, pueden definirse a partir de las anteriores, por ejemplo:

- Retrieval: dado un concepto C, retornar todos los individuos mencionados en la base de datos que son instancia de C.

$$\{a \in \mathrm{IND}(T) \mid T \models a{:}C\}$$

Tareas de Inferencia Complejas

Otras tareas de inferencia mas complejas, pueden definirse a partir de las anteriores, por ejemplo:

- Retrieval: dado un concepto C, retornar todos los individuos mencionados en la base de datos que son instancia de C.
  $$\{a \in \mathrm{IND}(T) \mid T \models a{:}C\}$$
- Conceptos mas específicos: dado un indivíduo a, retornar los conceptos más específicos en la ontología de los cuales a es un miembro.

Tareas de Inferencia Complejas

Otras tareas de inferencia mas complejas, pueden definirse a partir de las anteriores, por ejemplo:

- Retrieval: dado un concepto C, retornar todos los individuos mencionados en la base de datos que son instancia de C.
$$\{a \in \mathsf{IND}(T) \mid T \models a{:}C\}$$
- Conceptos mas específicos: dado un indivíduo a, retornar los conceptos más específicos en la ontología de los cuales a es un miembro.
- Conceptos parientes (descendientes) inmediatos: dado un concepto C, retornar los conceptos inmediatamente sobre (bajo) C en la jerarquía.

Relación con LPO

Relación con LPO

La mayoría de los DLs son fragmentos decidibles de LPO.

Relación con LPO

La mayoría de los DLs son fragmentos decidibles de LPO.

Tomar un lenguaje de PO que tenga

- un predicado unario C por cada concepto atómico C
- un predicado binario R por cada rol R
- una constante a por cada indivíduo a.

Relación con LPO

La mayoría de los DLs son fragmentos decidibles de LPO.

Tomar un lenguaje de PO que tenga

- un predicado unario C por cada concepto atómico C
- un predicado binario R por cada rol R
- una constante a por cada indivíduo a.

Definimos las siguientes traducciones $t_x : \mathcal{ALC} \to LPO$ y $t_y : \mathcal{ALC} \to LPO$ por recursión mutua

$$
\begin{aligned}
t_x(C) &= C(x) & \qquad t_y(C) &= C(y) \\
t_x(\neg C) &= \neg t_x(C) & \qquad t_y(\neg C) &= \neg t_y(C) \\
t_x(C \sqcap D) &= t_x(C) \wedge t_x(D) & \qquad t_y(C \sqcap D) &= t_y(C) \wedge t_y(D)
\end{aligned}
$$

Relación con LPO

La mayoría de los DLs son fragmentos decidibles de LPO.

Tomar un lenguaje de PO que tenga

- un predicado unario C por cada concepto atómico C
- un predicado binario R por cada rol R
- una constante a por cada indivíduo a.

Definimos las siguientes traducciones$t_x : \mathcal{ALC} \rightarrow LPO$ y $t_y : \mathcal{ALC} \rightarrow LPO$ por recursión mutua

$$
\begin{array}{rcl}
t_x(C) & = & C(x) \\
t_x(\neg C) & = & \neg t_x(C) \\
t_x(C \sqcap D) & = & t_x(C) \wedge t_x(D) \\
t_x(\exists R.C) & = & \exists y.(R(x,y) \wedge t_y(C))
\end{array}
\qquad
\begin{array}{rcl}
t_y(C) & = & C(y) \\
t_y(\neg C) & = & \neg t_y(C) \\
t_y(C \sqcap D) & = & t_y(C) \wedge t_y(D) \\
t_y(\exists R.C) & = & \exists x.(R(y,x) \wedge t_x(C))
\end{array}
$$

Traduciendo Bases de Conocimiento

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Una A-Box $A = \{a{:}C_i \mid i \leq n\} \cup \{(a,b){:}R_i \mid i \leq m\}$ se traduce como

$$t(A) = \bigwedge_{1 \leq i \leq n} t_x(C_i)[x/a] \wedge \bigwedge_{1 \leq i \leq m} R_i(a,b)$$

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Una A-Box $A = \{a{:}C_i \mid i \leq n\} \cup \{(a,b){:}R_i \mid i \leq m\}$ se traduce como

$$t(A) = \bigwedge_{1 \leq i \leq n} t_x(C_i)[x/a] \wedge \bigwedge_{1 \leq i \leq m} R_i(a,b)$$

T: C es satisfacible sii $t_x(C)$ es satisfacible.

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Una A-Box $A = \{a{:}C_i \mid i \leq n\} \cup \{(a, b){:}R_i \mid i \leq m\}$ se traduce como

$$t(A) = \bigwedge_{1 \leq i \leq n} t_x(C_i)[x/a] \wedge \bigwedge_{1 \leq i \leq m} R_i(a, b)$$

T: C es satisfacible sii $t_x(C)$ es satisfacible.
T: C es satisfacible respecto de $\langle T, A \rangle$ sii $t_x(C) \wedge t(T) \wedge t(A)$ es satisfacible.

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Una A-Box $A = \{a{:}C_i \mid i \leq n\} \cup \{(a, b){:}R_i \mid i \leq m\}$ se traduce como

$$t(A) = \bigwedge_{1 \leq i \leq n} t_x(C_i)[x/a] \wedge \bigwedge_{1 \leq i \leq m} R_i(a, b)$$

T: C es satisfacible sii $t_x(C)$ es satisfacible.
T: C es satisfacible respecto de $\langle T, A \rangle$ sii $t_x(C) \wedge t(T) \wedge t(A)$ es satisfacible.
T: C es subsumido por D sii $\forall x.(t_x(C) \rightarrow t_x(D))$ es válido.

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Una A-Box $A = \{a{:}C_i \mid i \leq n\} \cup \{(a, b){:}R_i \mid i \leq m\}$ se traduce como

$$t(A) = \bigwedge_{1 \leq i \leq n} t_x(C_i)[x/a] \wedge \bigwedge_{1 \leq i \leq m} R_i(a, b)$$

T: C es satisfacible sii $t_x(C)$ es satisfacible.
T: C es satisfacible respecto de $\langle T, A \rangle$ sii $t_x(C) \wedge t(T) \wedge t(A)$ es satisfacible.
T: C es subsumido por D sii $\forall x.(t_x(C) \rightarrow t_x(D))$ es válido.
T: C es subsumido por D respecto de $\langle T, A \rangle$ sii ...

Traduciendo Bases de Conocimiento

Una T-Box $T = \{C_i \sqsubseteq D_i \mid i \leq n\}$ se traduce como

$$t(T) = \forall x.(\bigwedge_{1 \leq i \leq n} t_x(C_i) \rightarrow t_x(D_i))$$

Una A-Box $A = \{a{:}C_i \mid i \leq n\} \cup \{(a,b){:}R_i \mid i \leq m\}$ se traduce como

$$t(A) = \bigwedge_{1 \leq i \leq n} t_x(C_i)[x/a] \wedge \bigwedge_{1 \leq i \leq m} R_i(a,b)$$

T: C es satisfacible sii $t_x(C)$ es satisfacible.

T: C es satisfacible respecto de $\langle T, A\rangle$ sii $t_x(C) \wedge t(T) \wedge t(A)$ es satisfacible.

T: C es subsumido por D sii $\forall x.(t_x(C) \rightarrow t_x(D))$ es válido.

T: C es subsumido por D respecto de $\langle T, A\rangle$ sii ...
$(t(T) \wedge t(A)) \rightarrow \forall x.(t_x(C) \rightarrow t_x(D))$ es válido.

Otros Operadores / Constraints

Otros Operadores / Constraints

	Number restriction ($\mathcal{N}$)
$(\leq n\ R)$	$\{d_1 \mid \|\|\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\|\| \leq n\}$
$(\geq n\ R)$	$\{d_1 \mid \|\|\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\|\| \geq n\}$

Otros Operadores / Constraints

Number restriction ($\mathcal{N}$)

$$
\begin{array}{ll}
(\leq n\ R) & \{d_1 \mid ||\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}|| \leq n\} \\
(\geq n\ R) & \{d_1 \mid ||\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}|| \geq n\}
\end{array}
$$

Qualified number restrictions ($\mathcal{Q}$)

$$
\begin{array}{ll}
(\leq n\ R\ C) & \{d_1 \mid ||\{d_2 \mid R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}}\}|| \leq n\} \\
(\geq n\ R\ C) & \{d_1 \mid ||\{d_2 \mid R^{\mathcal{I}}(d_1, d_2) \wedge C^{\mathcal{I}}\}|| \geq n\}
\end{array}
$$

Otros Operadores / Constraints

Number restriction ($\mathcal{N}$)

$(\leq n\ R)$	$\{d_1 \mid \|\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\| \leq n\}$
$(\geq n\ R)$	$\{d_1 \mid \|\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\| \geq n\}$

Qualified number restrictions ($\mathcal{Q}$)

$(\leq n\ R\ C)$	$\{d_1 \mid \|\{d_2 \mid R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}}\}\| \leq n\}$
$(\geq n\ R\ C)$	$\{d_1 \mid \|\{d_2 \mid R^{\mathcal{I}}(d_1, d_2) \wedge C^{\mathcal{I}}\}\| \geq n\}$

One-Of ($\mathcal{O}$)

$\{a_1, \ldots a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Otros Operadores / Constraints

Otros Operadores / Constraints

	Inverse roles ($\mathcal{I}$)
R^{-}	$\{(i,j) \mid (j,i) \in R^{\mathcal{I}}\}$

Otros Operadores / Constraints

	Inverse roles ($\mathcal{I}$)
R^-	$\{(i,j) \mid (j,i) \in R^{\mathcal{I}}\}$
	Role Intersection ($\mathcal{R}$)
$R \sqcap S$	$\{(i,j) \mid (i,j) \in R^{\mathcal{I}} \wedge (i,j) \in S^{\mathcal{I}}\}$

Otros Operadores / Constraints

	Inverse roles ($\mathcal{I}$)
R^-	$\{(i,j) \mid (j,i) \in R^\mathcal{I}\}$
	Role Intersection ($\mathcal{R}$)
$R \sqcap S$	$\{(i,j) \mid (i,j) \in R^\mathcal{I} \wedge (i,j) \in S^\mathcal{I}\}$
	Roles transitivos ($\mathcal{R}^+$)
$R = R^+$	$R^\mathcal{I}$ es una relación transitiva

Otros Operadores / Constraints

	Inverse roles ($\mathcal{I}$)
R^-	$\{(i,j) \mid (j,i) \in R^{\mathcal{I}}\}$
	Role Intersection ($\mathcal{R}$)
$R \sqcap S$	$\{(i,j) \mid (i,j) \in R^{\mathcal{I}} \wedge (i,j) \in S^{\mathcal{I}}\}$

	Roles transitivos ($\mathcal{R}^+$)
$R = R^+$	$R^{\mathcal{I}}$ es una relación transitiva
	Roles funcionales ($\mathcal{F}$)
R feature	$R^{\mathcal{I}}$ es una función (función parcial)

Otros Operadores / Constraints

	Inverse roles ($\mathcal{I}$)
R^-	$\{(i,j) \mid (j,i) \in R^{\mathcal{I}}\}$
	Role Intersection ($\mathcal{R}$)
$R \sqcap S$	$\{(i,j) \mid (i,j) \in R^{\mathcal{I}} \wedge (i,j) \in S^{\mathcal{I}}\}$

	Roles transitivos ($\mathcal{R}^+$)
$R = R^+$	$R^{\mathcal{I}}$ es una relación transitiva
	Roles funcionales ($\mathcal{F}$)
R feature	$R^{\mathcal{I}}$ es una función (función parcial)
	Jerarquía de Roles ($\mathcal{H}$)
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

DLs y Lógicas Modales

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.
- Podríamos decir que DLs son ‘el lado computacional de ML’ y que ML son ‘el lado teórico’ de DLs.

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.
- Podríamos decir que DLs son 'el lado computacional de ML' y que ML son 'el lado teórico' de DLs.

$$
\begin{array}{rcl}
\mathcal{ALC} & \Longleftrightarrow & \mathbf{K}_m \text{ (}\mathbf{K}\text{ multimodal)} \\
\neg C & \Longleftrightarrow & \neg C \\
C \sqcap D & \Longleftrightarrow & C \wedge D \\
\exists R.C & \Longleftrightarrow & \langle R \rangle C
\end{array}
$$

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.
- Podríamos decir que DLs son 'el lado computacional de ML' y que ML son 'el lado teórico' de DLs.

$$
\begin{array}{rcl}
\mathcal{ALC} & \Longleftrightarrow & \mathbf{K}_m \text{ } (\mathbf{K} \text{ multimodal}) \\
\neg C & \Longleftrightarrow & \neg C \\
C \sqcap D & \Longleftrightarrow & C \wedge D \\
\exists R.C & \Longleftrightarrow & \langle R \rangle C
\end{array}
$$

$$\text{roles transitivos} \quad \Longleftrightarrow \quad \text{frame transitivos } (\mathbf{K4})$$

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.
- Podríamos decir que DLs son 'el lado computacional de ML' y que ML son 'el lado teórico' de DLs.

$$
\begin{array}{rcl}
\mathcal{ALC} & \Longleftrightarrow & \mathbf{K}_m \text{ (}\mathbf{K}\text{ multimodal)} \\
\neg C & \Longleftrightarrow & \neg C \\
C \sqcap D & \Longleftrightarrow & C \wedge D \\
\exists R.C & \Longleftrightarrow & \langle R \rangle C
\end{array}
$$

$$
\begin{array}{rcl}
\text{roles transitivos} & \Longleftrightarrow & \text{frame transitivos (}\mathbf{K4}\text{)} \\
\text{expr. regulares sobre roles} & \Longleftrightarrow & \text{propositional dynamic logic (PDL)}
\end{array}
$$

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.
- Podríamos decir que DLs son 'el lado computacional de ML' y que ML son 'el lado teórico' de DLs.

$$
\begin{array}{rcl}
\mathcal{ALC} & \Longleftrightarrow & \mathbf{K}_m \text{ (}\mathbf{K}\text{ multimodal)} \\
\neg C & \Longleftrightarrow & \neg C \\
C \sqcap D & \Longleftrightarrow & C \wedge D \\
\exists R.C & \Longleftrightarrow & \langle R \rangle C
\end{array}
$$

$$
\begin{array}{rcl}
\text{roles transitivos} & \Longleftrightarrow & \text{frame transitivos (}\mathbf{K4}\text{)} \\
\text{expr. regulares sobre roles} & \Longleftrightarrow & \text{propositional dynamic logic (PDL)} \\
\text{roles inversos} & \Longleftrightarrow & \text{lógicas temporales (}\mathbf{K}_t\text{)}
\end{array}
$$

DLs y Lógicas Modales

- DLs y las lógicas modales (ML) son formalismos muy próximos.
- Podríamos decir que DLs son 'el lado computacional de ML' y que ML son 'el lado teórico' de DLs.

$$
\begin{array}{rcl}
\mathcal{ALC} & \iff & \mathbf{K}_m \text{ } (\mathbf{K} \text{ multimodal}) \\
\neg C & \iff & \neg C \\
C \sqcap D & \iff & C \wedge D \\
\exists R.C & \iff & \langle R \rangle C
\end{array}
$$

roles transitivos	$\iff$	frame transitivos (**K4**)
expr. regulares sobre roles	$\iff$	propositional dynamic logic (PDL)
roles inversos	$\iff$	lógicas temporales ($\mathbf{K}_t$)
number restrictions	$\iff$	graded modalities ($\Diamond_n \varphi$)

Complejidad Vía Traducción

Complejidad Vía Traducción

Sabiendo que muchas DLs son fragmentos de LPO (vía la traducción t), podemos transferir resultados conocidos sobre complejidad de estos fragmentos a ciertos lenguajes de descripción.

Complejidad Vía Traducción

Sabiendo que muchas DLs son fragmentos de LPO (vía la traducción t), podemos transferir resultados conocidos sobre complejidad de estos fragmentos a ciertos lenguajes de descripción.

- $\mathcal{ALC}$ es un fragmento de LPO con sólo dos variables (LPO^2) que se sabe decidible.

Complejidad Vía Traducción

Sabiendo que muchas DLs son fragmentos de LPO (vía la traducción t), podemos transferir resultados conocidos sobre complejidad de estos fragmentos a ciertos lenguajes de descripción.

- $\mathcal{ALC}$ es un fragmento de LPO con sólo dos variables (LPO^2) que se sabe decidible.
- Más aun, $\mathcal{ALC}$ con roles inversos y operadores Booleanos sobre roles está todavía en LPO^2!

Complejidad Vía Traducción

Sabiendo que muchas DLs son fragmentos de LPO (vía la traducción t), podemos transferir resultados conocidos sobre complejidad de estos fragmentos a ciertos lenguajes de descripción.

- $\mathcal{ALC}$ es un fragmento de LPO con sólo dos variables (LPO^2) que se sabe decidible.
- Más aun, $\mathcal{ALC}$ con roles inversos y operadores Booleanos sobre roles está todavía en LPO^2!
- Que pasa si agregamos $\mathcal{Q}$?

Complejidad Vía Traducción

Sabiendo que muchas DLs son fragmentos de LPO (vía la traducción t), podemos transferir resultados conocidos sobre complejidad de estos fragmentos a ciertos lenguajes de descripción.

- $\mathcal{ALC}$ es un fragmento de LPO con sólo dos variables (LPO^2) que se sabe decidible.
- Más aun, $\mathcal{ALC}$ con roles inversos y operadores Booleanos sobre roles está todavía en LPO^2!
- Que pasa si agregamos $\mathcal{Q}$? Aunque nos salimos de LPO^2, caemos en C^2: LPO^2 extendida con 'counting quantifiers' ($\exists^n x.\varphi(x)$ es 'existen n indivíduos diferentes en el dominio que satisfacen φ'), también decidible

Lower Bounds vs. Upper Bounds

Lower Bounds vs. Upper Bounds

- Usamos la traducción t para obtener Upper Bounds (i.e., el problema P no es más complejo que el problema Q.)

Lower Bounds vs. Upper Bounds

- Usamos la traducción t para obtener Upper Bounds (i.e., el problema P no es más complejo que el problema Q.)
- Pero no sirve para obtener Lower Bounds (i.e., el problema P es al menos tan complejo como el problema Q.). Para esto necesitaríamos traduccións 'para el otro lado'.

Lower Bounds vs. Upper Bounds

- Usamos la traducción t para obtener Upper Bounds (i.e., el problema P no es más complejo que el problema Q.)
- Pero no sirve para obtener Lower Bounds (i.e., el problema P es al menos tan complejo como el problema Q.). Para esto necesitaríamos traduccións 'para el otro lado'.
- Y en realidad, muchas veces las DLs tienen mejor comportamiento computacional que los fragmentos de LPO clásicos en los que pueden traducirse naturalmente.

Lower Bounds vs. Upper Bounds

- Usamos la traducción t para obtener Upper Bounds (i.e., el problema P no es más complejo que el problema Q.)
- Pero no sirve para obtener Lower Bounds (i.e., el problema P es al menos tan complejo como el problema Q.). Para esto necesitaríamos traduccións 'para el otro lado'.
- Y en realidad, muchas veces las DLs tienen mejor comportamiento computacional que los fragmentos de LPO clásicos en los que pueden traducirse naturalmente.E.g.,
  - Contrastando con muchas DLs, agregar 'roles transitivos' (requerir que ciertas relaciones binarias sean interpretadas como relaciones transitivas) a LPO^2 vuelve el fragmento indecidible.

Lower Bounds vs. Upper Bounds

- Usamos la traducción t para obtener Upper Bounds (i.e., el problema P no es más complejo que el problema Q.)
- Pero no sirve para obtener Lower Bounds (i.e., el problema P es al menos tan complejo como el problema Q.). Para esto necesitaríamos traduccións 'para el otro lado'.
- Y en realidad, muchas veces las DLs tienen mejor comportamiento computacional que los fragmentos de LPO clásicos en los que pueden traducirse naturalmente.E.g.,
  - Contrastando con muchas DLs, agregar 'roles transitivos' (requerir que ciertas relaciones binarias sean interpretadas como relaciones transitivas) a LPO^2 vuelve el fragmento indecidible.
  - LPO^2 es NExpTime-complete, mientras que la muchas DLs estan en ExpTime (o aún por debajo).

Lower Bounds vs. Upper Bounds

- Usamos la traducción t para obtener Upper Bounds (i.e., el problema P no es más complejo que el problema Q.)
- Pero no sirve para obtener Lower Bounds (i.e., el problema P es al menos tan complejo como el problema Q.). Para esto necesitaríamos traduccións 'para el otro lado'.
- Y en realidad, muchas veces las DLs tienen mejor comportamiento computacional que los fragmentos de LPO clásicos en los que pueden traducirse naturalmente.E.g.,
  - Contrastando con muchas DLs, agregar 'roles transitivos' (requerir que ciertas relaciones binarias sean interpretadas como relaciones transitivas) a LPO^2 vuelve el fragmento indecidible.
  - LPO^2 es NExpTime-complete, mientras que la muchas DLs estan en ExpTime (o aún por debajo).

Tableaux para $\mathcal{ALC}$

Tableaux para $\mathcal{ALC}$

- Miremos a los modelos de DL: Grafos etiquetados.
  Podemos pensar que un modelo de DL es
  - un conjunto de valuaciones proposicionales
  - más estructura relacional entre estas valuaciones

Tableaux para $\mathcal{ALC}$

- Miremos a los modelos de DL: Grafos etiquetados.
  Podemos pensar que un modelo de DL es
  - un conjunto de valuaciones proposicionales
  - más estructura relacional entre estas valuaciones
- Un tableau para DL entonces es
  - una colección de tableau proposicionales
  - con estructura adicional: la relacion de accesibilidad.

Tableaux para $\mathcal{ALC}$

- Miremos a los modelos de DL: Grafos etiquetados.
  Podemos pensar que un modelo de DL es
  - un conjunto de valuaciones proposicionales
  - más estructura relacional entre estas valuaciones
- Un tableau para DL entonces es
  - una colección de tableau proposicionales
  - con estructura adicional: la relacion de accesibilidad.
- Notar que esta es exactamente la información que hay en una ABox.

Tableaux para $\mathcal{ALC}$

- Miremos a los modelos de DL: Grafos etiquetados. Podemos pensar que un modelo de DL es
  - un conjunto de valuaciones proposicionales
  - más estructura relacional entre estas valuaciones
- Un tableau para DL entonces es
  - una colección de tableau proposicionales
  - con estructura adicional: la relacion de accesibilidad.
- Notar que esta es exactamente la información que hay en una ABox.
- Para determinar si un concepto C de $\mathcal{ALC}$ es consistente, escribir C en NNF ($(\neg\exists R.C) \leadsto (\forall R.\neg C)$ y $(\neg\forall R.C) \leadsto (\exists R.\neg C)$).

Tableaux para $\mathcal{ALC}$

- Miremos a los modelos de DL: Grafos etiquetados. Podemos pensar que un modelo de DL es
  - un conjunto de valuaciones proposicionales
  - más estructura relacional entre estas valuaciones
- Un tableau para DL entonces es
  - una colección de tableau proposicionales
  - con estructura adicional: la relacion de accesibilidad.
- Notar que esta es exactamente la información que hay en una ABox.
- Para determinar si un concepto C de $\mathcal{ALC}$ es consistente, escribir C en NNF $((\neg\exists R.C) \rightsquigarrow (\forall R.\neg C)$ y $(\neg\forall R.C) \rightsquigarrow (\exists R.\neg C))$.
- Aplicar las siguientes reglas a $\mathcal{A} = \{a{:}\mathrm{NNF}(C)\}$.

Reglas de Tableau para DLs

Reglas de Tableau para DLs

$\rightarrow_{\sqcap}$	Si	1. $x{:}C_1 \sqcap C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \not\subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\sqcap} \mathcal{A} \cup \{x{:}C_1, x{:}C_2\}$

Reglas de Tableau para DLs

$\rightarrow_{\sqcap}$	Si	1. $x{:}C_1 \sqcap C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \not\subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\sqcap} \mathcal{A} \cup \{x{:}C_1, x{:}C_2\}$
$\rightarrow_{\sqcup}$	Si	1. $x{:}C_1 \sqcup C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \cap \mathcal{A} = \emptyset$
	entonces	$\mathcal{A} \rightarrow_{\sqcup} \mathcal{A} \cup \{x{:}D\}$ para algun $D \in \{C_1, C_2\}$

Reglas de Tableau para DLs

$\rightarrow_{\sqcap}$	Si	1. $x{:}C_1 \sqcap C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \not\subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\sqcap} \mathcal{A} \cup \{x{:}C_1, x{:}C_2\}$
$\rightarrow_{\sqcup}$	Si	1. $x{:}C_1 \sqcup C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \cap \mathcal{A} = \emptyset$
	entonces	$\mathcal{A} \rightarrow_{\sqcup} \mathcal{A} \cup \{x{:}D\}$ para algun $D \in \{C_1, C_2\}$
$\rightarrow_{\exists}$	Si	1. $x{:}\exists R.D \in \mathcal{A}$ y
		2. no hay y tq. $\{(x, y){:}R, y{:}D\} \subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\exists} \mathcal{A} \cup \{(x, y){:}R, y{:}D\}$ para y nuevo

Reglas de Tableau para DLs

$\rightarrow_{\sqcap}$	Si	1. $x{:}C_1 \sqcap C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \not\subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\sqcap} \mathcal{A} \cup \{x{:}C_1, x{:}C_2\}$
$\rightarrow_{\sqcup}$	Si	1. $x{:}C_1 \sqcup C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \cap \mathcal{A} = \emptyset$
	entonces	$\mathcal{A} \rightarrow_{\sqcup} \mathcal{A} \cup \{x{:}D\}$ para algun $D \in \{C_1, C_2\}$
$\rightarrow_{\exists}$	Si	1. $x{:}\exists R.D \in \mathcal{A}$ y
		2. no hay y tq. $\{(x, y){:}R, y{:}D\} \subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\exists} \mathcal{A} \cup \{(x, y){:}R, y{:}D\}$ para y nuevo
$\rightarrow_{\forall}$	si	1. $x{:}\forall R.D \in \mathcal{A}$ y
		2. hay un y tq. $(x, y){:}R \in \mathcal{A}$ and $y{:}D \notin \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\forall} \mathcal{A} \cup \{y{:}D\}$

Reglas de Tableau para DLs

$\rightarrow_{\sqcap}$	Si	1. $x{:}C_1 \sqcap C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \not\subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\sqcap} \mathcal{A} \cup \{x{:}C_1, x{:}C_2\}$
$\rightarrow_{\sqcup}$	Si	1. $x{:}C_1 \sqcup C_2 \in \mathcal{A}$ y
		2. $\{x{:}C_1, x{:}C_2\} \cap \mathcal{A} = \emptyset$
	entonces	$\mathcal{A} \rightarrow_{\sqcup} \mathcal{A} \cup \{x{:}D\}$ para algun $D \in \{C_1, C_2\}$
$\rightarrow_{\exists}$	Si	1. $x{:}\exists R.D \in \mathcal{A}$ y
		2. no hay y tq. $\{(x,y){:}R, y{:}D\} \subseteq \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\exists} \mathcal{A} \cup \{(x,y){:}R, y{:}D\}$ para y nuevo
$\rightarrow_{\forall}$	si	1. $x{:}\forall R.D \in \mathcal{A}$ y
		2. hay un y tq. $(x,y){:}R \in \mathcal{A}$ and $y{:}D \notin \mathcal{A}$
	entonces	$\mathcal{A} \rightarrow_{\forall} \mathcal{A} \cup \{y{:}D\}$

Clash: $\mathcal{A}$ tiene clash si para algun C, $\{a{:}C, a{:}\neg C\} \subseteq \mathcal{A}$.

Terminación

Terminación

Por que termina este algoritmo?

Terminación

Por que termina este algoritmo?

- Sea $\mathcal{L}(w) = \{C \mid w{:}C \in \mathcal{A}\}$.
- Las reglas $\sqcup$, $\sqcap$, $\exists$ pueden ser aplicadas sólo una vez a una fórmula en $\mathcal{L}(w)$

Terminación

Por que termina este algoritmo?

- Sea $\mathcal{L}(w) = \{C \mid w{:}C \in \mathcal{A}\}$.
- Las reglas $\sqcup$, $\sqcap$, $\exists$ pueden ser aplicadas sólo una vez a una fórmula en $\mathcal{L}(w)$
- La regla $\forall$ puede ser aplicada muchas veces a una formula en $\mathcal{L}(w)$ pero sólo una vez a cada eje (w, v).

Terminación

Por que termina este algoritmo?

- Sea $\mathcal{L}(w) = \{C \mid w{:}C \in \mathcal{A}\}$.
- Las reglas $\sqcup$, $\sqcap$, $\exists$ pueden ser aplicadas sólo una vez a una fórmula en $\mathcal{L}(w)$
- La regla $\forall$ puede ser aplicada muchas veces a una formula en $\mathcal{L}(w)$ pero sólo una vez a cada eje (w, v).
- Aplicar una regla a una fórmula φ extiende el labeling con una fórmula que es siempre estrictamente más pequeña que φ.

Correctitud y Completitud del Algoritmo

Correctitud y Completitud del Algoritmo

La correctitud y completitud del algoritmo se sigue de las siguientes propiedades:

1. No puede haber una sequencia infinita de aplicaciones de reglas (terminación)

Correctitud y Completitud del Algoritmo

La correctitud y completitud del algoritmo se sigue de las siguientes propiedades:

1. No puede haber una sequencia infinita de aplicaciones de reglas (terminación)
2. Sea $\mathcal{A}'$ obtenido a partir de $\mathcal{A}$ por la aplicación de alguna de las reglas. Entonces $\mathcal{A}'$ es satisfacible sii $\mathcal{A}$ es satisfacible.

Correctitud y Completitud del Algoritmo

La correctitud y completitud del algoritmo se sigue de las siguientes propiedades:

1. No puede haber una sequencia infinita de aplicaciones de reglas (terminación)
2. Sea $\mathcal{A}'$ obtenido a partir de $\mathcal{A}$ por la aplicación de alguna de las reglas. Entonces $\mathcal{A}'$ es satisfacible sii $\mathcal{A}$ es satisfacible.
3. Toda Abox $\mathcal{A}$ conteniendo un clash es insatisfacible.

Correctitud y Completitud del Algoritmo

La correctitud y completitud del algoritmo se sigue de las siguientes propiedades:

1. No puede haber una sequencia infinita de aplicaciones de reglas (terminación)
2. Sea $\mathcal{A}'$ obtenido a partir de $\mathcal{A}$ por la aplicación de alguna de las reglas. Entonces $\mathcal{A}'$ es satisfacible sii $\mathcal{A}$ es satisfacible.
3. Toda Abox $\mathcal{A}$ conteniendo un clash es insatisfacible.
4. Toda Abox $\mathcal{A}$ saturada (no nueva regla puede aplicarse a $\mathcal{A}$) y sin clash es satisfacible.

Que Regla Aplicamos Primero?

Que Regla Aplicamos Primero?

- Importa que regla aplicamos primero para decidibilidad?

Que Regla Aplicamos Primero?

- Importa que regla aplicamos primero para decidibilidad?
- Importa que regla aplicamos primero para tiempo de corrida?

Que Regla Aplicamos Primero?

- Importa que regla aplicamos primero para decidibilidad?
- Importa que regla aplicamos primero para tiempo de corrida?
- Algunas heurísticas para decidir que regla de expansión aplicar primero:

Que Regla Aplicamos Primero?

- Importa que regla aplicamos primero para decidibilidad?
- Importa que regla aplicamos primero para tiempo de corrida?
- Algunas heurísticas para decidir que regla de expansión aplicar primero:
  1. Usar non-branching rules (como $\sqcap$-rule) antes que branching rules (como $\sqcup$-rule)

Que Regla Aplicamos Primero?

- Importa que regla aplicamos primero para decidibilidad?
- Importa que regla aplicamos primero para tiempo de corrida?

- Algunas heurísticas para decidir que regla de expansión aplicar primero:
  1. Usar non-branching rules (como $\sqcap$-rule) antes que branching rules (como $\sqcup$-rule)
  2. usar reglas proposicionales (como $\sqcap$-rule y $\sqcup$-rule) antes que reglas modales (como $\exists$-rule y $\forall$-rule).

Complejidad del Algoritmo

Complejidad del Algoritmo

- El algoritmo que presentamos hasta el momento puede requerir espacio (y tiempo) exponencial!

Complejidad del Algoritmo

- El algoritmo que presentamos hasta el momento puede requerir espacio (y tiempo) exponencial!
- Consideremos el concepto definido recursivamente como

$$\begin{aligned} C_1 &:= \exists R.A \sqcap \exists R.B \\ C_{n+1} &:= \exists R.A \sqcap \exists R.B \sqcap \forall R.C_n \end{aligned}$$

Complejidad del Algoritmo

- El algoritmo que presentamos hasta el momento puede requerir espacio (y tiempo) exponencial!
- Consideremos el concepto definido recursivamente como

$$\begin{aligned} C_1 &:= \exists R.A \sqcap \exists R.B \\ C_{n+1} &:= \exists R.A \sqcap \exists R.B \sqcap \forall R.C_n \end{aligned}$$

- El tamaño de C_n es sólo lineal en n, pero el algoritmo de tableaux construiría, al ser corrido sobre $\mathcal{A} = \{a{:}C_n\}$ una ABox conteniendo $2^{n+1} - 1$ indivíduos.

Satisfacibilidad de $\mathcal{ALC}$ esta en PSPACE

Satisfacibilidad de $\mathcal{ALC}$ esta en PSPACE

- Toda fórmula satisfacible φ puede ser satisfecha en la raíz de un árbol finito de profundidad a lo sumo $\deg(\varphi) + 1$

Satisfacibilidad de $\mathcal{ALC}$ esta en PSPACE

- Toda fórmula satisfacible φ puede ser satisfecha en la raíz de un árbol finito de profundidad a lo sumo $\deg(\varphi) + 1$
- El tamaño total del modelo puede ser exponencial en $|\varphi|$, pero
  - no es necesario mantener toda esta informacion en memoria.
  - en cada momento, sólo necesitamos mantener la información correspondiente a una rama del modelo.

El Algoritmo

El Algoritmo

$\mathcal{ALC}\text{-SAT}(C) := \mathsf{sat}(x_0, \{x_0{:}C\})$

El Algoritmo

$\mathcal{ALC}\text{-SAT}(C) := \mathrm{sat}(x_0, \{x_0{:}C\})$

$\mathrm{sat}(x, \mathcal{A})$:
1. while ($\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ pueden aplicarse) y ($\mathcal{A}$ no tiene clash) do
2. aplicar $\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ a $\mathcal{A}$
3. if $\mathcal{A}$ tiene clash then return UNSAT.

El Algoritmo

$\mathcal{ALC}\text{-SAT}(C) := \mathsf{sat}(x_0, \{x_0{:}C\})$

sat($x, \mathcal{A}$):

1. while ($\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ pueden aplicarse) y ($\mathcal{A}$ no tiene clash) do
2. aplicar $\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ a $\mathcal{A}$
3. if $\mathcal{A}$ tiene clash then return UNSAT.
4. **E**:=$\{x{:}\exists R.D \mid x{:}\exists R.D \in \mathcal{A}\}$
5. while **E** $\neq \emptyset$ do
6. elegir $x{:}\exists R.D \in$ **E** arbitrario
7. $\mathcal{A}_{new} := \{(x, y){:}R, y{:}D\}$ donde y es un nuevo individuo.

El Algoritmo

$\mathcal{ALC}\text{-SAT}(C) := \text{sat}(x_0, \{x_0{:}C\})$

sat$(x, \mathcal{A})$:
1. while ($\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ pueden aplicarse) y ($\mathcal{A}$ no tiene clash) do
2.   aplicar $\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ a $\mathcal{A}$
3. if $\mathcal{A}$ tiene clash then return UNSAT.
4. **E**:=$\{x{:}\exists R.D \mid x{:}\exists R.D \in \mathcal{A}\}$
5. while **E** $\neq \emptyset$ do
6.   elegir $x{:}\exists R.D \in$ **E** arbitrario
7. $\mathcal{A}_{new} := \{(x, y){:}R, y{:}D\}$ donde y es un nuevo individuo.
8.   while ($\rightarrow_\forall$ puede aplicarse a $\mathcal{A} \cup \mathcal{A}_{new}$) do
9.   aplicar $\rightarrow_\forall$ a $\mathcal{A} \cup \mathcal{A}_{new}$
10.   if $\mathcal{A} \cup \mathcal{A}_{new}$ tiene clash then return UNSAT.

El Algoritmo

$\mathcal{ALC}\text{-SAT}(C) := \mathsf{sat}(x_0, \{x_0{:}C\})$

sat($x, \mathcal{A}$):
1. while ($\rightarrow_{\sqcap}$ o $\rightarrow_{\sqcup}$ pueden aplicarse) y ($\mathcal{A}$ no tiene clash) do
2.   aplicar $\rightarrow_{\sqcap}$ o $\rightarrow_{\sqcup}$ a $\mathcal{A}$
3. if $\mathcal{A}$ tiene clash then return UNSAT.
4. **E**:=$\{x{:}\exists R.D \mid x{:}\exists R.D \in \mathcal{A}\}$
5. while **E** $\neq \emptyset$ do
6.   elegir $x{:}\exists R.D \in$ **E** arbitrario
7. $\mathcal{A}_{new} := \{(x, y){:}R, y{:}D\}$ donde y es un nuevo individuo.
8.   while ($\rightarrow_{\forall}$ puede aplicarse a $\mathcal{A} \cup \mathcal{A}_{new}$) do
9.   aplicar $\rightarrow_{\forall}$ a $\mathcal{A} \cup \mathcal{A}_{new}$
10.   if $\mathcal{A} \cup \mathcal{A}_{new}$ tiene clash then return UNSAT.
11.   if sat($y, \mathcal{A} \cup \mathcal{A}_{new}$) = UNSAT then return UNSAT

El Algoritmo

$\mathcal{ALC}\text{-SAT}(C) := \text{sat}(x_0, \{x_0{:}C\})$

sat$(x, \mathcal{A})$:
1. while ($\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ pueden aplicarse) y ($\mathcal{A}$ no tiene clash) do
2.   aplicar $\rightarrow_\sqcap$ o $\rightarrow_\sqcup$ a $\mathcal{A}$
3. if $\mathcal{A}$ tiene clash then return UNSAT.
4. $\mathbf{E}{:=}\{x{:}\exists R.D \mid x{:}\exists R.D \in \mathcal{A}\}$
5. while $\mathbf{E} \neq \emptyset$ do
6.   elegir $x{:}\exists R.D \in \mathbf{E}$ arbitrario
7. $\mathcal{A}_{new} := \{(x, y){:}R, y{:}D\}$ donde y es un nuevo individuo.
8.   while ($\rightarrow_\forall$ puede aplicarse a $\mathcal{A} \cup \mathcal{A}_{new}$) do
9.   aplicar $\rightarrow_\forall$ a $\mathcal{A} \cup \mathcal{A}_{new}$
10.   if $\mathcal{A} \cup \mathcal{A}_{new}$ tiene clash then return UNSAT.
11.   if sat$(y, \mathcal{A} \cup \mathcal{A}_{new})$ = UNSAT then return UNSAT
12. $\mathbf{E} := \mathbf{E} \setminus \{x{:}\exists R.D\}$
13.   eliminar $\mathcal{A}_{new}$ de la memoria
14. return SAT

Terminologías

Terminologías

- El algoritmo que vimos hasta el momento sólo trata consistencia de conceptos

Terminologías

- El algoritmo que vimos hasta el momento sólo trata consistencia de conceptos
- Cómo agregamos ABoxes?

Terminologías

- El algoritmo que vimos hasta el momento sólo trata consistencia de conceptos
- Cómo agregamos ABoxes?
- Cómo agregamos TBoxes?
  - Consideremos una definición $C \sqsubseteq D$
  - Basta asegurarnos que cada nodo del tableaux contenga $\neg C \sqcup D$

Terminologías

- El algoritmo que vimos hasta el momento sólo trata consistencia de conceptos
- Cómo agregamos ABoxes?
- Cómo agregamos TBoxes?
  - Consideremos una definición $C \sqsubseteq D$
  - Basta asegurarnos que cada nodo del tableaux contenga $\neg C \sqcup D$
  - Pero eso implica que el tamaño de los conceptos en un nodo (y en particuar la profundidad máxima de cuantificación) no disminuye.

Terminologías

- El algoritmo que vimos hasta el momento sólo trata consistencia de conceptos
- Cómo agregamos ABoxes?
- Cómo agregamos TBoxes?
  - Consideremos una definición $C \sqsubseteq D$
  - Basta asegurarnos que cada nodo del tableaux contenga $\neg C \sqcup D$
  - Pero eso implica que el tamaño de los conceptos en un nodo (y en particuar la profundidad máxima de cuantificación) no disminuye.
  - El argumento anterior de terminación (y por lo tanto el de complejidad) se 'caen'.

Falla de Terminación: Terminologías

Falla de Terminación: Terminologías

- Como dijimos, el algoritmo básico de tableaux no termina para $\mathcal{ALC}$ con T-boxes generales.

Falla de Terminación: Terminologías

- Como dijimos, el algoritmo básico de tableaux no termina para $\mathcal{ALC}$ con T-boxes generales.
- E.g., si $\mathsf{Human} \sqsubseteq \exists\mathsf{has\text{-}mother.Human} \in T$, entonces $\neg\mathsf{Human} \sqcup \exists\mathsf{has\text{-}mother.Human}$ se agregaría a todo nodo del tableaux de $w{:}\mathsf{Human}$.

w $\quad \mathcal{L}(w) = \{\mathsf{Human}, (\neg\mathsf{Human} \sqcup \exists\mathsf{has\text{-}mother.Human}), \exists\mathsf{has\text{-}mother.Human}\}$

$\downarrow$ has-mother

x $\quad \mathcal{L}(x) = \{\mathsf{Human}, (\neg\mathsf{Human} \sqcup \exists\mathsf{has\text{-}mother.Human}), \exists\mathsf{has\text{-}mother.Human}\}$

$\downarrow$ has-mother

y $\quad \mathcal{L}(y) = \{\mathsf{Human}, (\neg\mathsf{Human} \sqcup \exists\mathsf{has\text{-}mother.Human}), \exists\mathsf{has\text{-}mother.Human}\}$

$\downarrow$

Blocking

Blocking

- Cuando un nuevo nodo es creado, chequear los ancestros por un etiqueta idéntica (o un superconjunto).

Blocking

- Cuando un nuevo nodo es creado, chequear los ancestros por un etiqueta idéntica (o un superconjunto).
- Si un nodo de este tipo existe, entonces el nuevo nodo esta bloqueado y ninguna regla puede aplicarse a el

$$\text{w} \quad \mathcal{L}(w) = \{\text{Human}, (\neg\text{Human} \sqcup \exists\text{has-mother.Human}), \exists\text{has-mother.Human}\}$$

$$\downarrow \text{has-mother}$$

$$\text{x} \quad \mathcal{L}(x) = \{\text{Human}\}$$

Implementaciones Naive

Implementaciones Naive

Problemas típicos

- Problemas de Espacio
  - Espacio requerido para las estructuras de datos que prepresentan el tableaux.
  - Raramente un problema serio en la práctica

Implementaciones Naive

Problemas típicos

- Problemas de Espacio
  - Espacio requerido para las estructuras de datos que prepresentan el tableaux.
  - Raramente un problema serio en la práctica
- Problemas de Tiempo
  - Necesitamos búsqueda dada la naturaleza no determinística del algoritmo de tableaux.
  - Un problema serio en la práctica
  - puede ser mitigado mediante
    - La elección cuidadosa del algoritmo
    - Una implementacion altamente optimizada

Tableaux para $\mathcal{I}$ y $\mathcal{N}$

Tableaux para $\mathcal{I}$ y $\mathcal{N}$

- Como modificamos el tableaux para $\mathcal{ALC}$ para que funcione también para $\mathcal{I}$?

Tableaux para $\mathcal{I}$ y $\mathcal{N}$

- Como modificamos el tableaux para $\mathcal{ALC}$ para que funcione también para $\mathcal{I}$?
- Reglas de Tableaux para $\mathcal{N}$

$\rightarrow_{\geq}$ Si 1. $x{:}(\leq nR) \in \mathcal{A}$ y

2. no hay individuos $z_1, \ldots, z_n$ tal que $(x, z_i){:}R \in \mathcal{A}$ y $z_i \dot{\neq} z_j \in \mathcal{A}$ $(1 \leq i < j \leq n)$

entonces $\mathcal{A} \rightarrow_{\geq} \mathcal{A} \cup \{(x, y_i){:}R \mid 1 \leq i \leq n\} \cup$
$\{y_i \dot{\neq} y_j \mid 1 \leq i < j \leq n\}$ para y_i nuevos

Tableaux para $\mathcal{I}$ y $\mathcal{N}$

- Como modificamos el tableaux para $\mathcal{ALC}$ para que funcione también para $\mathcal{I}$?
- Reglas de Tableaux para $\mathcal{N}$

$\rightarrow_{\geq}$ Si 1. $x{:}(\leq nR) \in \mathcal{A}$ y
2. no hay individuos $z_1, \ldots, z_n$ tal que $(x, z_i){:}R \in \mathcal{A}$ y $z_i \dot{\neq} z_j \in \mathcal{A}$ $(1 \leq i < j \leq n)$

entonces $\mathcal{A} \rightarrow_{\geq} \mathcal{A} \cup \{(x, y_i){:}R \mid 1 \leq i \leq n\} \cup \{y_i \dot{\neq} y_j \mid 1 \leq i < j \leq n\}$ para y_i nuevos

$\rightarrow_{\leq}$ Si 1. $x{:}(\leq nR) \in \mathcal{A}$ y
2. $\{(x, y_i){:}R \mid 1 \leq n \leq n + 1\} \subseteq \mathcal{A}$ and $y_i \dot{\neq} y_j \notin \mathcal{A}$ para algun i,j $1 \leq i < j \leq n + 1$

entonces $\mathcal{A} \rightarrow_{\leq} \mathcal{A}[y_i / y_j]$ para algun par $y_i \dot{\neq} y_j$

Tableaux para $\mathcal{I}$ y $\mathcal{N}$

- Como modificamos el tableaux para $\mathcal{ALC}$ para que funcione también para $\mathcal{I}$?
- Reglas de Tableaux para $\mathcal{N}$

$\rightarrow_{\geq}$ Si 1. $x{:}(\leq nR) \in \mathcal{A}$ y
2. no hay individuos $z_1, \dots, z_n$ tal que $(x, z_i){:}R \in \mathcal{A}$ y $z_i \dot{\neq} z_j \in \mathcal{A}$ $(1 \leq i < j \leq n)$
entonces $\mathcal{A} \rightarrow_{\geq} \mathcal{A} \cup \{(x, y_i){:}R \mid 1 \leq i \leq n\} \cup$ $\{y_i \dot{\neq} y_j \mid 1 \leq i < j \leq n\}$ para y_i nuevos

$\rightarrow_{\leq}$ Si 1. $x{:}(\leq nR) \in \mathcal{A}$ y
2. $\{(x, y_i){:}R \mid 1 \leq n \leq n+1\} \subseteq \mathcal{A}$ and $y_i \dot{\neq} y_j \notin \mathcal{A}$ para algun i,j $1 \leq i < j \leq n+1$
entonces $\mathcal{A} \rightarrow_{\leq} \mathcal{A}[y_i/y_j]$ para algun par $y_i \dot{\neq} y_j$

Clash (para $\mathcal{N}$): $\mathcal{A}$ tiene un clash también si $\{x{:}(\leq nR)\} \cup$ $\{(x, y_i) : R \mid 1 \leq i \leq n+1\} \cup \{y_i \dot{\neq} y_j \mid 1 \leq i < j \leq n+1\} \subseteq \mathcal{A}$.

System Demo: RACER

System Demo: RACER

- RACER (http://www.sts.tu-harburg.de/ r.f.moeller/racer/), desarrollado en la University of Hamburg por Haarslev y Möller en Common Lisp. Puede trabajar en $\mathcal{ALCFHIQ}(\mathcal{D}^-)_{\mathcal{R}^+}$.

System Demo: RACER

- RACER (http://www.sts.tu-harburg.de/ r.f.moeller/racer/), desarrollado en la University of Hamburg por Haarslev y Möller en Common Lisp. Puede trabajar en $\mathcal{ALCFHIQ}(\mathcal{D}^-)_{\mathcal{R}^+}$.
- RACER es un razonador automático para DL con soporte para TBoxes, ABoxes y Concrete domains (e.g., (in)ecuaciones lineares sobre los reales)

System Demo: RACER

- RACER (http://www.sts.tu-harburg.de/ r.f.moeller/racer/), desarrollado en la University of Hamburg por Haarslev y Möller en Common Lisp. Puede trabajar en $\mathcal{ALCFHIQ}(\mathcal{D}^-)_{\mathcal{R}^+}$.
- RACER es un razonador automático para DL con soporte para TBoxes, ABoxes y Concrete domains (e.g., (in)ecuaciones lineares sobre los reales)
- Es también una herramienta de inferencia para la web semántica que permite el desarrollo de ontologías, consultas de documentos RDF y ontologias RDFS/DAML que permite el registro permanente de queries con notificación automática de nuevos resultados

System Demo: RACER

- RACER (http://www.sts.tu-harburg.de/ r.f.moeller/racer/), desarrollado en la University of Hamburg por Haarslev y Möller en Common Lisp. Puede trabajar en $\mathcal{ALCFHIQ}(\mathcal{D}^-)_{\mathcal{R}^+}$.
- RACER es un razonador automático para DL con soporte para TBoxes, ABoxes y Concrete domains (e.g., (in)ecuaciones lineares sobre los reales)
- Es también una herramienta de inferencia para la web semántica que permite el desarrollo de ontologías, consultas de documentos RDF y ontologias RDFS/DAML que permite el registro permanente de queries con notificación automática de nuevos resultados
- Está implementado en Lisp y existen versiones para Linux, Macintosh y Windows