Description Logic 1: Syntax and Semantics

Leif Harald Karlsen

Autumn 2016

Contents

Contents

Overview

– *Description logics* are formal languages designed for knowledge representation and reasoning, and most of these are decidable fragments of FOL.

Overview

- *Description logics* are formal languages designed for knowledge representation and reasoning, and most of these are decidable fragments of FOL.
- Each description logic describes a language, and each language differ in expressibility vs. reasoning complexity, defined by allowing or disallowing different constructs (e.g. conjunction, disjunction, negation, quantifiers, etc.) in their language.

Overview

- *Description logics* are formal languages designed for knowledge representation and reasoning, and most of these are decidable fragments of FOL.
- Each description logic describes a language, and each language differ in expressibility vs. reasoning complexity, defined by allowing or disallowing different constructs (e.g. conjunction, disjunction, negation, quantifiers, etc.) in their language.
- Expressiveness: Propositional logic $\rightarrow$ Description logics $\rightarrow$ First order logic

History and motivation

– Description logic comes from a merging of two traditions.

History and motivation

- Description logic comes from a merging of two traditions.
- Knowledge Representation (KR)
  - Application oriented
  - Represent ‘knowledge’ in some way
  - ‘Frames,’ like classes, with relations and attributes
  - Try to add some ‘semantics’ in order to do some ‘reasoning’

History and motivation

- Description logic comes from a merging of two traditions.
- Knowledge Representation (KR)
  - Application oriented
  - Represent ‘knowledge’ in some way
  - ‘Frames,’ like classes, with relations and attributes
  - Try to add some ‘semantics’ in order to do some ‘reasoning’
- Automated Reasoning, Modal Logic
  - Had theorems and algorithms

History and motivation

- Description logic comes from a merging of two traditions.
- Knowledge Representation (KR)
  - Application oriented
  - Represent ‘knowledge’ in some way
  - ‘Frames,’ like classes, with relations and attributes
  - Try to add some ‘semantics’ in order to do some ‘reasoning’
- Automated Reasoning, Modal Logic
  - Had theorems and algorithms
- Cross-fertilisation of applications and theory

History and motivation

- Description logic comes from a merging of two traditions.
- Knowledge Representation (KR)
  - Application oriented
  - Represent 'knowledge' in some way
  - 'Frames,' like classes, with relations and attributes
  - Try to add some 'semantics' in order to do some 'reasoning'
- Automated Reasoning, Modal Logic
  - Had theorems and algorithms
- Cross-fertilisation of applications and theory
- Today: large impact on Semantic Web (sign up for INF3580/4580!)

Knowledge bases

In description logics one works with three different types of elements:

Knowledge bases

In description logics one works with three different types of elements:

– individuals/constants (e.g. *james*, *sensor*1)

Knowledge bases

In description logics one works with three different types of elements:

- individuals/constants (e.g. james, $\mathit{sensor}1$)
- concepts/unary relations (e.g. Person, Sensor)

Knowledge bases

In description logics one works with three different types of elements:

- individuals/constants (e.g. *james*, $sensor1$)
- concepts/unary relations (e.g. *Person*, *Sensor*)
- roles/binary relations (e.g. *isFatherOf*, *isConnectedTo*)

Knowledge bases

In description logics one works with three different types of elements:

- individuals/constants (e.g. *james*, $sensor1$)
- concepts/unary relations (e.g. *Person*, *Sensor*)
- roles/binary relations (e.g. *isFatherOf*, *isConnectedTo*)

Knowledge is represented as a *knowledge base*, $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ where:

- $\mathcal{A}$ is a set of *assertions* about named individuals, called the *ABox* (e.g. $Person(james)$, $isFatherOf(james, peter)$)

Knowledge bases

In description logics one works with three different types of elements:

- individuals/constants (e.g. *james*, $sensor1$)
- concepts/unary relations (e.g. *Person*, *Sensor*)
- roles/binary relations (e.g. *isFatherOf*, *isConnectedTo*)

Knowledge is represented as a *knowledge base*, $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ where:

- $\mathcal{A}$ is a set of *assertions* about named individuals, called the *ABox* (e.g. $Person(james)$, $isFatherOf(james, peter)$)
- $\mathcal{T}$ is a set of terminology definitions (i.e. complex descriptions of concepts or roles), called the *TBox* (e.g. $Human \sqsubseteq Mammal$, $Mother \equiv Parent \sqcap Woman$)

Contents

$\mathcal{ALC}$: Syntax

The description logic $\mathcal{ALC}$ (Attribute Language with general Complement) allows the following concepts:

$\mathcal{ALC}$: Syntax

The description logic $\mathcal{ALC}$ (Attribute Language with general Complement) allows the following concepts:

$$
\begin{array}{lll}
C, D \rightarrow & A & \mid \quad \text{(atomic concept)} \\
& \top & \mid \quad \text{(universal concept)} \\
& \bot & \mid \quad \text{(bottom concept)} \\
& \neg C & \mid \quad \text{(negation)} \\
& C \sqcup D & \mid \quad \text{(union)} \\
& C \sqcap D & \mid \quad \text{(intersection)} \\
& \exists R.C & \mid \quad \text{(existential restriction)} \\
& \forall R.C & \mid \quad \text{(universal restriction)}
\end{array}
$$

$\mathcal{ALC}$: Syntax

The description logic $\mathcal{ALC}$ (Attribute Language with general Complement) allows the following concepts:

$$
\begin{array}{llll}
C, D \rightarrow & A & \mid & \text{(atomic concept)} \\
& \top & \mid & \text{(universal concept)} \\
& \bot & \mid & \text{(bottom concept)} \\
& \neg C & \mid & \text{(negation)} \\
& C \sqcup D & \mid & \text{(union)} \\
& C \sqcap D & \mid & \text{(intersection)} \\
& \exists R.C & \mid & \text{(existential restriction)} \\
& \forall R.C & \mid & \text{(universal restriction)}
\end{array}
$$

where A is an atomic concept, C and D are concepts, and R is a role.

$\mathcal{ALC}$: Syntax

The description logic $\mathcal{ALC}$ (Attribute Language with general Complement) allows the following concepts:

$$
\begin{array}{rlcl}
C, D \rightarrow & A & \mid & \text{(atomic concept)} \\
& \top & \mid & \text{(universal concept)} \\
& \bot & \mid & \text{(bottom concept)} \\
& \neg C & \mid & \text{(negation)} \\
& C \sqcup D & \mid & \text{(union)} \\
& C \sqcap D & \mid & \text{(intersection)} \\
& \exists R.C & \mid & \text{(existential restriction)} \\
& \forall R.C & \mid & \text{(universal restriction)}
\end{array}
$$

where A is an atomic concept, C and D are concepts, and R is a role. We allow

- ABox assertions: $C(a)$ and $R(a, b)$ for individuals a, b, concepts C and roles R;
- TBox axioms: $C \sqsubseteq D$ for concepts C and D.

Examples

Complex concepts:

$$C \sqcap (D \sqcup E)$$
$$\exists R.(A \sqcap B)$$
$$\neg \forall R.\exists P.(A \sqcap \exists Q.\top)$$

Examples

Complex concepts:

$$C \sqcap (D \sqcup E)$$
$$\exists R.(A \sqcap B)$$
$$\neg \forall R.\exists P.(A \sqcap \exists Q.\top)$$

TBox axioms:

$$\begin{aligned} C \sqcup D &\sqsubseteq E \\ \exists R.A &\sqsubseteq B \\ F \sqcap G &\sqsubseteq \bot \\ H &\sqsubseteq \forall P.I \\ L \sqcap J &\sqsubseteq \neg K \\ \exists P.B &\sqsubseteq \neg \forall R.\exists P.(A \sqcap \exists Q.\top) \end{aligned}$$

$\mathcal{ALC}$: Semantics

A model $\mathcal{M}$ for a knowledge base $\mathcal{K}$ consists of

- a nonempty set Δ, and
- an interpretation function $\_^{\mathcal{M}}$, such that:
  - for every constant c, $c^{\mathcal{M}} \in \Delta$,
  - for every atomic concept A, $A^{\mathcal{M}} \subseteq \Delta$,
  - for every atomic role R, $R^{\mathcal{M}} \subseteq \Delta \times \Delta$,

$\mathcal{ALC}$: Semantics

$\_^{\mathcal{M}}$ is extended inductively as

$\mathcal{ALC}$: Semantics

$\_^{\mathcal{M}}$ is extended inductively as

$$
\begin{aligned}
\top^{\mathcal{M}} &= \Delta \\
\bot^{\mathcal{M}} &= \emptyset \\
(\neg C)^{\mathcal{M}} &= \Delta \setminus C^{\mathcal{M}} \\
(C \sqcup D)^{\mathcal{M}} &= C^{\mathcal{M}} \cup D^{\mathcal{M}} \\
(C \sqcap D)^{\mathcal{M}} &= C^{\mathcal{M}} \cap D^{\mathcal{M}} \\
(\forall R.C)^{\mathcal{M}} &= \left\{ a \in \Delta \mid \forall b \in \Delta \left(\langle a, b \rangle \in R^{\mathcal{M}} \to b \in C^{\mathcal{M}} \right) \right\} \\
(\exists R.C)^{\mathcal{M}} &= \left\{ a \in \Delta \mid \exists b \in \Delta \left(\langle a, b \rangle \in R^{\mathcal{M}} \wedge b \in C^{\mathcal{M}} \right) \right\}
\end{aligned}
$$

$\mathcal{ALC}$: Semantics

An interpretation $\mathcal{M}$ satisfies

- $C(a)$, denoted $\mathcal{M} \vDash C(a)$, iff $a^{\mathcal{M}} \in C^{\mathcal{M}}$;

$\mathcal{ALC}$: Semantics

An interpretation $\mathcal{M}$ satisfies

- $C(a)$, denoted $\mathcal{M} \vDash C(a)$, iff $a^{\mathcal{M}} \in C^{\mathcal{M}}$;
- $C \sqsubseteq D$, denoted $\mathcal{M} \vDash C \sqsubseteq D$, iff $C^{\mathcal{M}} \subseteq D^{\mathcal{M}}$;

$\mathcal{ALC}$: Semantics

An interpretation $\mathcal{M}$ satisfies

- $C(a)$, denoted $\mathcal{M} \vDash C(a)$, iff $a^{\mathcal{M}} \in C^{\mathcal{M}}$;
- $C \sqsubseteq D$, denoted $\mathcal{M} \vDash C \sqsubseteq D$, iff $C^{\mathcal{M}} \subseteq D^{\mathcal{M}}$;
- $R \sqsubseteq P$, denoted $\mathcal{M} \vDash R \sqsubseteq P$, iff $R^{\mathcal{M}} \subseteq P^{\mathcal{M}}$.

$\mathcal{ALC}$: Semantics

An interpretation $\mathcal{M}$ satisfies

- $C(a)$, denoted $\mathcal{M} \vDash C(a)$, iff $a^{\mathcal{M}} \in C^{\mathcal{M}}$;
- $C \sqsubseteq D$, denoted $\mathcal{M} \vDash C \sqsubseteq D$, iff $C^{\mathcal{M}} \subseteq D^{\mathcal{M}}$;
- $R \sqsubseteq P$, denoted $\mathcal{M} \vDash R \sqsubseteq P$, iff $R^{\mathcal{M}} \subseteq P^{\mathcal{M}}$.

As usual, we will write $\mathcal{K} \vDash \psi$ if for any model $\mathcal{M}$ we have that $\mathcal{M} \vDash \mathcal{K} \Rightarrow \mathcal{M} \vDash \psi$.

$\mathcal{ALC}$: Semantics

An interpretation $\mathcal{M}$ satisfies

- $C(a)$, denoted $\mathcal{M} \vDash C(a)$, iff $a^{\mathcal{M}} \in C^{\mathcal{M}}$;
- $C \sqsubseteq D$, denoted $\mathcal{M} \vDash C \sqsubseteq D$, iff $C^{\mathcal{M}} \subseteq D^{\mathcal{M}}$;
- $R \sqsubseteq P$, denoted $\mathcal{M} \vDash R \sqsubseteq P$, iff $R^{\mathcal{M}} \subseteq P^{\mathcal{M}}$.

As usual, we will write $\mathcal{K} \vDash \psi$ if for any model $\mathcal{M}$ we have that $\mathcal{M} \vDash \mathcal{K} \Rightarrow \mathcal{M} \vDash \psi$.

We will use the following shorthand notation:

- $C \equiv D$ instead of the two axioms $C \sqsubseteq D$ and $D \sqsubseteq C$;
- $\mathcal{A} \vDash \psi$ instead of $\langle \emptyset, \mathcal{A} \rangle \vDash \psi$;
- $\mathcal{T} \vDash \psi$ instead of $\langle \mathcal{T}, \emptyset \rangle \vDash \psi$.

Example

Assume $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{T}$ is the TBox:

$$
\begin{aligned}
Animal &\sqsubseteq LivingThing \\
Donkey &\equiv Animal \sqcap \forall hasParent.Donkey \\
Horse &\equiv Animal \sqcap \forall hasParent.Horse \\
Mule &\equiv Animal \sqcap \exists hasParent.Horse \sqcap \exists hasParent.Donkey \\
\exists hasParent.Mule &\sqsubseteq \bot
\end{aligned}
$$

Example

Assume $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{T}$ is the TBox:

$$
\begin{aligned}
\mathit{Animal} &\sqsubseteq \mathit{LivingThing} \\
\mathit{Donkey} &\equiv \mathit{Animal} \sqcap \forall \mathit{hasParent}.\mathit{Donkey} \\
\mathit{Horse} &\equiv \mathit{Animal} \sqcap \forall \mathit{hasParent}.\mathit{Horse} \\
\mathit{Mule} &\equiv \mathit{Animal} \sqcap \exists \mathit{hasParent}.\mathit{Horse} \sqcap \exists \mathit{hasParent}.\mathit{Donkey} \\
\exists \mathit{hasParent}.\mathit{Mule} &\sqsubseteq \bot
\end{aligned}
$$

---

and $\mathcal{A}$ is the ABox:

$$
\mathit{Horse}(\mathit{mary}) \quad \mathit{Horse}(\mathit{peter}) \quad \mathit{Donkey}(\mathit{sven}) \quad \mathit{Animal}(\mathit{hannah}) \quad \mathit{Animal}(\mathit{carl})
$$

$$
\begin{array}{ll}
\mathit{hasParent}(\mathit{hannah}, \mathit{mary}) & \mathit{hasParent}(\mathit{hannah}, \mathit{sven}) \\
\mathit{hasParent}(\mathit{carl}, \mathit{mary}) & \mathit{hasParent}(\mathit{carl}, \mathit{peter})
\end{array}
$$

Example

Assume $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{T}$ is the TBox:

$$
\begin{aligned}
Animal &\sqsubseteq LivingThing \\
Donkey &\equiv Animal \sqcap \forall hasParent.Donkey \\
Horse &\equiv Animal \sqcap \forall hasParent.Horse \\
Mule &\equiv Animal \sqcap \exists hasParent.Horse \sqcap \exists hasParent.Donkey \\
\exists hasParent.Mule &\sqsubseteq \bot
\end{aligned}
$$

---

and $\mathcal{A}$ is the ABox:

$$
Horse(mary) \quad Horse(peter) \quad Donkey(sven) \quad Animal(hannah) \quad Animal(carl)
$$

$$
\begin{array}{ll}
hasParent(hannah, mary) & hasParent(hannah, sven) \\
hasParent(carl, mary) & hasParent(carl, peter)
\end{array}
$$

Then we have $\mathcal{K} \models Mule(hannah)$, but $\mathcal{K} \not\models Horse(carl)$.

Translation to First order logic

The function π map concepts to first-order formulae:

Translation to First order logic

The function π map concepts to first-order formulae:

$$
\begin{aligned}
\pi_x(A) &= A(x) \\
\pi_x(\neg C) &= \neg \pi_x(C) \\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D) \\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D) \\
\pi_x(\exists R.C) &= \exists y \, (R(x,y) \wedge \pi_y(C)) \\
\pi_x(\forall R.C) &= \forall y \, (R(x,y) \to \pi_y(C))
\end{aligned}
$$

Translation to First order logic

The function π map concepts to first-order formulae:

$$\begin{aligned}
\pi_x(A) &= A(x) \\
\pi_x(\neg C) &= \neg \pi_x(C) \\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D) \\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D) \\
\pi_x(\exists R.C) &= \exists y\,(R(x,y) \wedge \pi_y(C)) \\
\pi_x(\forall R.C) &= \forall y\,(R(x,y) \rightarrow \pi_y(C))
\end{aligned}$$

We can then map axioms: $\Pi(C \sqsubseteq D) := \forall x(\pi_x(C) \rightarrow \pi_x(D))$.

Translation to First order logic

The function π map concepts to first-order formulae:

$$\begin{aligned}
\pi_x(A) &= A(x)\\
\pi_x(\neg C) &= \neg\pi_x(C)\\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D)\\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D)\\
\pi_x(\exists R.C) &= \exists y\,(R(x,y) \wedge \pi_y(C))\\
\pi_x(\forall R.C) &= \forall y\,(R(x,y) \rightarrow \pi_y(C))
\end{aligned}$$

We can then map axioms: $\Pi(C \sqsubseteq D) := \forall x(\pi_x(C) \rightarrow \pi_x(D))$.

Theorem

$a^{\mathcal{I}} \in C^{\mathcal{I}}$ iff $\mathcal{I} \models_{FOL} \pi_x(C)[a/x]$, and $\mathcal{I} \vDash C \sqsubseteq D$ iff $\mathcal{I} \models_{FOL} \Pi(C \sqsubseteq D)$.

Translation to First order logic

The function π map concepts to first-order formulae:

$$\begin{aligned}
\pi_x(A) &= A(x)\\
\pi_x(\neg C) &= \neg\pi_x(C)\\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D)\\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D)\\
\pi_x(\exists R.C) &= \exists y\,(R(x,y) \wedge \pi_y(C))\\
\pi_x(\forall R.C) &= \forall y\,(R(x,y) \rightarrow \pi_y(C))
\end{aligned}$$

We can then map axioms: $\Pi(C \sqsubseteq D) := \forall x(\pi_x(C) \rightarrow \pi_x(D))$.

Theorem

$a^{\mathcal{I}} \in C^{\mathcal{I}}$ iff $\mathcal{I} \models_{FOL} \pi_x(C)[a/x]$, and $\mathcal{I} \vDash C \sqsubseteq D$ iff $\mathcal{I} \models_{FOL} \Pi(C \sqsubseteq D)$.

E.g.:

$$\begin{aligned}
\pi_x\,(Animal \sqcap \forall hasParent.Donkey) &= Animal(x) \wedge \forall y(hasParent(x,y) \rightarrow Donkey(y))\\
\Pi(Animal \sqsubseteq LivingThing) &= \forall x(Animal(x) \rightarrow LivingThing(x))
\end{aligned}$$

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

– Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\}\rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);
- Given two concepts C and D, are C and D equivalent ($\mathcal{T} \models C \equiv D$);

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);
- Given two concepts C and D, are C and D equivalent ($\mathcal{T} \models C \equiv D$);
- Given two concepts C and D, are C and D disjoint ($\mathcal{T} \models C \sqcap D \sqsubseteq \bot$);

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);
- Given two concepts C and D, are C and D equivalent ($\mathcal{T} \models C \equiv D$);
- Given two concepts C and D, are C and D disjoint ($\mathcal{T} \models C \sqcap D \sqsubseteq \bot$);

The following problems are of interest with respect to knowledge bases $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$:

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);
- Given two concepts C and D, are C and D equivalent ($\mathcal{T} \models C \equiv D$);
- Given two concepts C and D, are C and D disjoint ($\mathcal{T} \models C \sqcap D \sqsubseteq \bot$);

The following problems are of interest with respect to knowledge bases $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$:

- Is $\mathcal{K}$ consistent ($\mathcal{K}$ has a model);

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);
- Given two concepts C and D, are C and D equivalent ($\mathcal{T} \models C \equiv D$);
- Given two concepts C and D, are C and D disjoint ($\mathcal{T} \models C \sqcap D \sqsubseteq \bot$);

The following problems are of interest with respect to knowledge bases $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$:

- Is $\mathcal{K}$ consistent ($\mathcal{K}$ has a model);
- Given a concept C and an individual a, does $\mathcal{K}$ entail $C(a)$ ($\mathcal{K} \models C(a)$);

Reasoning problems

The following problems are of interest with respect to a TBox $\mathcal{T}$:

- Given a concept C, is C satisfiable ($\langle \mathcal{T}, \{C(x_0)\} \rangle$ has a model);
- Given two concepts C and D, is C subsumed by D ($\mathcal{T} \models C \sqsubseteq D$);
- Given two concepts C and D, are C and D equivalent ($\mathcal{T} \models C \equiv D$);
- Given two concepts C and D, are C and D disjoint ($\mathcal{T} \models C \sqcap D \sqsubseteq \bot$);

The following problems are of interest with respect to knowledge bases $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$:

- Is $\mathcal{K}$ consistent ($\mathcal{K}$ has a model);
- Given a concept C and an individual a, does $\mathcal{K}$ entail $C(a)$ ($\mathcal{K} \models C(a)$);
- Given a concept C, find all individuals a such that $\mathcal{K}$ entails $C(a)$.

Contents

Naming conventions

– As we have seen $\mathcal{ALC}$ is the *Attribute Language with general Complement*.

Naming conventions

- As we have seen $\mathcal{ALC}$ is the *Attribute Language with general Complement*.
- The $\mathcal{C}$ actually denotes an extension of a more restrictive language $\mathcal{AL}$.

Naming conventions

- As we have seen $\mathcal{ALC}$ is the *Attribute Language with general Complement*.
- The $\mathcal{C}$ actually denotes an extension of a more restrictive language $\mathcal{AL}$.
- In a similar way, we have the following possible extensions of our logic:

Naming conventions

- As we have seen $\mathcal{ALC}$ is the *Attribute Language with general Complement*.
- The $\mathcal{C}$ actually denotes an extension of a more restrictive language $\mathcal{AL}$.
- In a similar way, we have the following possible extensions of our logic:
  - $\mathcal{H}$: Role hierarchies;
  - $\mathcal{R}$: Complex role hierarchies;
  - $\mathcal{N}$: Cardinality restrictions;
  - $\mathcal{Q}$: Qualified cardinality restrictions;
  - $\mathcal{O}$: Closed classes;
  - $\mathcal{I}$: Inverse roles;
  - $\mathcal{D}$: Datatypes;
  - ...

Naming conventions

- As we have seen $\mathcal{ALC}$ is the *Attribute Language with general Complement*.
- The $\mathcal{C}$ actually denotes an extension of a more restrictive language $\mathcal{AL}$.
- In a similar way, we have the following possible extensions of our logic:
  - $\mathcal{H}$: Role hierarchies;
  - $\mathcal{R}$: Complex role hierarchies;
  - $\mathcal{N}$: Cardinality restrictions;
  - $\mathcal{Q}$: Qualified cardinality restrictions;
  - $\mathcal{O}$: Closed classes;
  - $\mathcal{I}$: Inverse roles;
  - $\mathcal{D}$: Datatypes;
  - ...
- We name the languages by adding the letters of the features to $\mathcal{ALC}$. So e.g. $\mathcal{ALCN}$ is $\mathcal{ALC}$ extended with cardinality restrictions and $\mathcal{ALCHI}$ is $\mathcal{ALC}$ extended with role hierarchies and inverse roles.

Naming conventions

- As we have seen $\mathcal{ALC}$ is the *Attribute Language with general Complement*.
- The $\mathcal{C}$ actually denotes an extension of a more restrictive language $\mathcal{AL}$.
- In a similar way, we have the following possible extensions of our logic:
  - $\mathcal{H}$: Role hierarchies;
  - $\mathcal{R}$: Complex role hierarchies;
  - $\mathcal{N}$: Cardinality restrictions;
  - $\mathcal{Q}$: Qualified cardinality restrictions;
  - $\mathcal{O}$: Closed classes;
  - $\mathcal{I}$: Inverse roles;
  - $\mathcal{D}$: Datatypes;
  - ...
- We name the languages by adding the letters of the features to $\mathcal{ALC}$. So e.g. $\mathcal{ALCN}$ is $\mathcal{ALC}$ extended with cardinality restrictions and $\mathcal{ALCHI}$ is $\mathcal{ALC}$ extended with role hierarchies and inverse roles.
- It is common to shorten $\mathcal{ALC}$ (extended with transitive roles) to just $\mathcal{S}$ for more advanced languages, so e.g. $\mathcal{SHOIN}$ is $\mathcal{ALC} + \mathcal{H} + \mathcal{O} + \mathcal{I} + \mathcal{N}$.

Normal extensions

– $\mathcal{H}$ – Role Hierarchies: We allow TBox axioms on the form $R \sqsubseteq P$ for atomic roles. Semantics:

$$\mathcal{M} \models R \sqsubseteq P \Leftrightarrow R^{\mathcal{M}} \subseteq P^{\mathcal{M}}$$

e.g. $hasFather \sqsubseteq hasParent$;

Normal extensions

- $\mathcal{H}$ – Role Hierarchies: We allow TBox axioms on the form $R \sqsubseteq P$ for atomic roles. Semantics:

  $$\mathcal{M} \vDash R \sqsubseteq P \Leftrightarrow R^{\mathcal{M}} \subseteq P^{\mathcal{M}}$$

  e.g. *hasFather* $\sqsubseteq$ *hasParent*;

- $\mathcal{R}$ – Complex role hierarchies: We allow roles on the form $R \circ P$ and TBox axioms on the form $R \circ P \sqsubseteq P$ and $R \circ P \sqsubseteq R$ for any two roles. Semantics:

  $$(R \circ P)^{\mathcal{M}} := \{\langle a, b\rangle \in \Delta^{\mathcal{M}} \times \Delta^{\mathcal{M}} \mid \exists c \in \Delta^{\mathcal{M}} (\langle a, c\rangle \in R^{\mathcal{M}} \wedge \langle c, b\rangle \in P^{\mathcal{M}})\}$$

  and

  $$\mathcal{M} \vDash R \sqsubseteq P \Leftrightarrow R^{\mathcal{M}} \subseteq P^{\mathcal{M}}$$

  e.g. *friendOf* $\circ$ *enemyOf* $\sqsubseteq$ *enemyOf*.

Normal extensions

– $\mathcal{N}$ – Cardinality restrictions: We allow concepts on the form $\leq n\, R$ and $\geq n\, R$ for any natural number n. Semantics[1]:

$$(\leq n\, R)^{\mathcal{M}} := \{a \in \Delta^{\mathcal{M}} \mid \#\{b \in \Delta^{\mathcal{M}} \mid \langle a, b\rangle \in R^{\mathcal{M}}\} \leq n\}$$
$$(\geq n\, R)^{\mathcal{M}} := \{a \in \Delta^{\mathcal{M}} \mid \#\{b \in \Delta^{\mathcal{M}} \mid \langle a, b\rangle \in R^{\mathcal{M}}\} \geq n\}$$

e.g. $Mammal \sqsubseteq\ \leq 2\, hasParent$;

[1] We let $\#S$ be the cardinality of the set S

Normal extensions

- $\mathcal{N}$ – Cardinality restrictions: We allow concepts on the form $\leq n\,R$ and $\geq n\,R$ for any natural number n. Semantics[1]:

$$(\leq n\,R)^{\mathcal{M}} := \{a \in \Delta^{\mathcal{M}} \mid \#\{b \in \Delta^{\mathcal{M}} \mid \langle a, b \rangle \in R^{\mathcal{M}}\} \leq n\}$$
$$(\geq n\,R)^{\mathcal{M}} := \{a \in \Delta^{\mathcal{M}} \mid \#\{b \in \Delta^{\mathcal{M}} \mid \langle a, b \rangle \in R^{\mathcal{M}}\} \geq n\}$$

  e.g. $\mathit{Mammal} \sqsubseteq\ \leq 2\,\mathit{hasParent}$;

- $\mathcal{Q}$ – Qualified cardinality restrictions: We allow concepts on the form $\leq n\,R.C$ and $\geq n\,R.C$ for any natural number n. Semantics:

$$(\leq n\,R.C)^{\mathcal{M}} := \{a \in \Delta^{\mathcal{M}} \mid \#\{b \in \Delta^{\mathcal{M}} \mid \langle a, b \rangle \in R^{\mathcal{M}} \wedge b \in C^{\mathcal{M}}\} \leq n\}$$
$$(\geq n\,R.C)^{\mathcal{M}} := \{a \in \Delta^{\mathcal{M}} \mid \#\{b \in \Delta^{\mathcal{M}} \mid \langle a, b \rangle \in R^{\mathcal{M}} \wedge b \in C^{\mathcal{M}}\} \geq n\}$$

  e.g. $\geq 2\,\mathit{owns}.\mathit{House} \sqsubseteq \mathit{RichPeople}$.

[1] We let $\#S$ be the cardinality of the set S

Normal extensions

– $\mathcal{O}$ – Closed classes: We allow concepts on the form $\{a_1, a_2, \dots, a_n\}$ where a_i are individuals. Semantics

$$(\{a_1, a_2, \dots, a_n\})^{\mathcal{M}} := \{a_1^{\mathcal{M}}, a_2^{\mathcal{M}}, \dots, a_n^{\mathcal{M}}\}$$

e.g. $Days \equiv \{monday, tuesday, wednesday, thursday, friday, saturday, sunday\}$;

Normal extensions

- $\mathcal{O}$ – Closed classes: We allow concepts on the form $\{a_1, a_2, \ldots, a_n\}$ where a_i are individuals. Semantics

  $$(\{a_1, a_2, \ldots, a_n\})^{\mathcal{M}} := \{a_1^{\mathcal{M}}, a_2^{\mathcal{M}}, \ldots, a_n^{\mathcal{M}}\}$$

  e.g. $Days \equiv \{monday, tuesday, wednesday, thursday, friday, saturday, sunday\}$;
- $\mathcal{I}$ – Inverse roles: We allow roles on the form R^-. Semantics:

  $$(R^-)^{\mathcal{M}} := \{\langle a, b\rangle \in \Delta^{\mathcal{M}} \times \Delta^{\mathcal{M}} \mid \langle b, a\rangle \in R^{\mathcal{M}}\}$$

  e.g. $hasParent^- \sqsubseteq hasChild$;

Normal extensions

- $\mathcal{O}$ – Closed classes: We allow concepts on the form $\{a_1, a_2, \ldots, a_n\}$ where a_i are individuals. Semantics

  $$(\{a_1, a_2, \ldots, a_n\})^{\mathcal{M}} := \{a_1^{\mathcal{M}}, a_2^{\mathcal{M}}, \ldots, a_n^{\mathcal{M}}\}$$

  e.g. $\mathit{Days} \equiv \{\mathit{monday}, \mathit{tuesday}, \mathit{wednesday}, \mathit{thursday}, \mathit{friday}, \mathit{saturday}, \mathit{sunday}\}$;
- $\mathcal{I}$ – Inverse roles: We allow roles on the form R^-. Semantics:

  $$(R^-)^{\mathcal{M}} := \{\langle a, b\rangle \in \Delta^{\mathcal{M}} \times \Delta^{\mathcal{M}} \mid \langle b, a\rangle \in R^{\mathcal{M}}\}$$

  e.g. $\mathit{hasParent}^- \sqsubseteq \mathit{hasChild}$;
- $\mathcal{D}$ - Datatypes: We introduce a set of datatypes: *int*, *string*, *float*, *boolean*, and so on. They all have a fixed interpretation, that is, the same for all models.

Examples

$$\mathit{OnlyChild} \quad \sqsubseteq \quad \mathit{Person} \sqcap \neg \exists \mathit{hasSibling}.\top$$

Examples

$$
\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg \exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal}
\end{array}
$$

Examples

$$
\begin{aligned}
OnlyChild &\sqsubseteq Person \sqcap \neg\exists hasSibling.\top \\
Animal &\sqsubseteq\ \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person &\sqsubseteq \bot
\end{aligned}
$$

Examples

$$\begin{aligned}
\mathit{OnlyChild} &\sqsubseteq \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} &\sqsubseteq\ \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} &\sqsubseteq \bot \\
\mathit{Person} &\sqsubseteq \exists \mathit{loves}.\{\mathit{mary}\}
\end{aligned}$$

Examples

$$
\begin{aligned}
OnlyChild &\sqsubseteq Person \sqcap \neg \exists hasSibling.\top \\
Animal &\sqsubseteq \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person &\sqsubseteq \bot \\
Person &\sqsubseteq \exists loves.\{mary\} \\
Norwegian &\sqsubseteq \exists comesFrom.\{norway\}
\end{aligned}
$$

Examples

$$
\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg \exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} & \sqsubseteq & \bot \\
\mathit{Person} & \sqsubseteq & \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} & \sqsubseteq & \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} & \sqsubseteq & \neg\{\mathit{eve}\}
\end{array}
$$

Examples

$$\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} & \sqsubseteq & \bot \\
\mathit{Person} & \sqsubseteq & \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} & \sqsubseteq & \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} & \sqsubseteq & \neg\{\mathit{eve}\} \\
\mathit{hasFather} \circ \mathit{hasBrother} & \sqsubseteq & \mathit{hasUncle}
\end{array}$$

Examples

$$
\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} & \sqsubseteq & \bot \\
\mathit{Person} & \sqsubseteq & \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} & \sqsubseteq & \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} & \sqsubseteq & \neg\{\mathit{eve}\} \\
\mathit{hasFather} \circ \mathit{hasBrother} & \sqsubseteq & \mathit{hasUncle} \\
\\
\exists R.\top & \sqsubseteq & C
\end{array}
$$

Examples

$$
\begin{aligned}
\mathit{OnlyChild} &\sqsubseteq \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} &\sqsubseteq\ \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} &\sqsubseteq \bot \\
\mathit{Person} &\sqsubseteq \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} &\sqsubseteq \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} &\sqsubseteq \neg\{\mathit{eve}\} \\
\mathit{hasFather} \circ \mathit{hasBrother} &\sqsubseteq \mathit{hasUncle}
\end{aligned}
$$

$$\exists R.\top \sqsubseteq C \qquad\qquad \text{Domain}$$

Examples

$$
\begin{aligned}
OnlyChild &\sqsubseteq Person \sqcap \neg\exists hasSibling.\top \\
Animal &\sqsubseteq \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person &\sqsubseteq \bot \\
Person &\sqsubseteq \exists loves.\{mary\} \\
Norwegian &\sqsubseteq \exists comesFrom.\{norway\} \\
\{adam\} &\sqsubseteq \neg\{eve\} \\
hasFather \circ hasBrother &\sqsubseteq hasUncle
\end{aligned}
$$

$$
\begin{aligned}
\exists R.\top &\sqsubseteq C \qquad\qquad \text{Domain} \\
\top &\sqsubseteq \forall R.C
\end{aligned}
$$

Examples

$$
\begin{array}{rcll}
OnlyChild & \sqsubseteq & Person \sqcap \neg\exists hasSibling.\top & \\
Animal & \sqsubseteq & \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal & \\
Pet \sqcap Person & \sqsubseteq & \bot & \\
Person & \sqsubseteq & \exists loves.\{mary\} & \\
Norwegian & \sqsubseteq & \exists comesFrom.\{norway\} & \\
\{adam\} & \sqsubseteq & \neg\{eve\} & \\
hasFather \circ hasBrother & \sqsubseteq & hasUncle & \\
 & & & \\
\exists R.\top & \sqsubseteq & C & \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \text{Range}
\end{array}
$$

Examples

$$
\begin{array}{rcl}
OnlyChild & \sqsubseteq & Person \sqcap \neg\exists hasSibling.\top \\
Animal & \sqsubseteq & \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person & \sqsubseteq & \bot \\
Person & \sqsubseteq & \exists loves.\{mary\} \\
Norwegian & \sqsubseteq & \exists comesFrom.\{norway\} \\
\{adam\} & \sqsubseteq & \neg\{eve\} \\
hasFather \circ hasBrother & \sqsubseteq & hasUncle
\end{array}
$$

$$
\begin{array}{rcll}
\exists R.\top & \sqsubseteq & C & \qquad\qquad \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \qquad\qquad \text{Range} \\
R \circ R & \sqsubseteq & R &
\end{array}
$$

Examples

$$
\begin{aligned}
OnlyChild &\sqsubseteq Person \sqcap \neg\exists hasSibling.\top \\
Animal &\sqsubseteq \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person &\sqsubseteq \bot \\
Person &\sqsubseteq \exists loves.\{mary\} \\
Norwegian &\sqsubseteq \exists comesFrom.\{norway\} \\
\{adam\} &\sqsubseteq \neg\{eve\} \\
hasFather \circ hasBrother &\sqsubseteq hasUncle
\end{aligned}
$$

$$
\begin{aligned}
\exists R.\top &\sqsubseteq C && \text{Domain} \\
\top &\sqsubseteq \forall R.C && \text{Range} \\
R \circ R &\sqsubseteq R && \text{Transitivity}
\end{aligned}
$$

Examples

$$
\begin{array}{rcl}
OnlyChild & \sqsubseteq & Person \sqcap \neg\exists hasSibling.\top \\
Animal & \sqsubseteq & \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person & \sqsubseteq & \bot \\
Person & \sqsubseteq & \exists loves.\{mary\} \\
Norwegian & \sqsubseteq & \exists comesFrom.\{norway\} \\
\{adam\} & \sqsubseteq & \neg\{eve\} \\
hasFather \circ hasBrother & \sqsubseteq & hasUncle
\end{array}
$$

$$
\begin{array}{rcll}
\exists R.\top & \sqsubseteq & C & \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \text{Range} \\
R \circ R & \sqsubseteq & R & \text{Transitivity} \\
\top & \sqsubseteq & \leq 1\, R.\top &
\end{array}
$$

Examples

$$
\begin{array}{rcl}
OnlyChild & \sqsubseteq & Person \sqcap \neg\exists hasSibling.\top \\
Animal & \sqsubseteq & \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person & \sqsubseteq & \bot \\
Person & \sqsubseteq & \exists loves.\{mary\} \\
Norwegian & \sqsubseteq & \exists comesFrom.\{norway\} \\
\{adam\} & \sqsubseteq & \neg\{eve\} \\
hasFather \circ hasBrother & \sqsubseteq & hasUncle
\end{array}
$$

$$
\begin{array}{rcll}
\exists R.\top & \sqsubseteq & C & \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \text{Range} \\
R \circ R & \sqsubseteq & R & \text{Transitivity} \\
\top & \sqsubseteq & \leq 1\, R.\top & \text{Functionality}
\end{array}
$$

Examples

$$\begin{array}{rcl}
OnlyChild & \sqsubseteq & Person \sqcap \neg\exists hasSibling.\top \\
Animal & \sqsubseteq & \leq 2\ hasParent.Animal \sqcap \geq 2\ hasParent.Animal \\
Pet \sqcap Person & \sqsubseteq & \bot \\
Person & \sqsubseteq & \exists loves.\{mary\} \\
Norwegian & \sqsubseteq & \exists comesFrom.\{norway\} \\
\{adam\} & \sqsubseteq & \neg\{eve\} \\
hasFather \circ hasBrother & \sqsubseteq & hasUncle
\end{array}$$

$$\begin{array}{rcll}
\exists R.\top & \sqsubseteq & C & \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \text{Range} \\
R \circ R & \sqsubseteq & R & \text{Transitivity} \\
\top & \sqsubseteq & \leq 1\, R.\top & \text{Functionality} \\
R & \sqsubseteq & R^{-} &
\end{array}$$

Examples

$$
\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} & \sqsubseteq & \bot \\
\mathit{Person} & \sqsubseteq & \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} & \sqsubseteq & \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} & \sqsubseteq & \neg\{\mathit{eve}\} \\
\mathit{hasFather} \circ \mathit{hasBrother} & \sqsubseteq & \mathit{hasUncle}
\end{array}
$$

$$
\begin{array}{rcll}
\exists R.\top & \sqsubseteq & C & \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \text{Range} \\
R \circ R & \sqsubseteq & R & \text{Transitivity} \\
\top & \sqsubseteq & \leq 1\, R.\top & \text{Functionality} \\
R & \sqsubseteq & R^{-} & \text{Symmetry}
\end{array}
$$

Examples

$$
\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} & \sqsubseteq & \bot \\
\mathit{Person} & \sqsubseteq & \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} & \sqsubseteq & \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} & \sqsubseteq & \neg\{\mathit{eve}\} \\
\mathit{hasFather} \circ \mathit{hasBrother} & \sqsubseteq & \mathit{hasUncle}
\end{array}
$$

$\exists R.\top$	$\sqsubseteq$	C	Domain
$\top$	$\sqsubseteq$	$\forall R.C$	Range
$R \circ R$	$\sqsubseteq$	R	Transitivity
$\top$	$\sqsubseteq$	$\leq 1\, R.\top$	Functionality
R	$\sqsubseteq$	R^-	Symmetry
R	$\sqsubseteq$	$\neg R^-$	

Examples

$$\begin{array}{rcl}
\mathit{OnlyChild} & \sqsubseteq & \mathit{Person} \sqcap \neg\exists \mathit{hasSibling}.\top \\
\mathit{Animal} & \sqsubseteq & \leq 2\ \mathit{hasParent}.\mathit{Animal} \sqcap \geq 2\ \mathit{hasParent}.\mathit{Animal} \\
\mathit{Pet} \sqcap \mathit{Person} & \sqsubseteq & \bot \\
\mathit{Person} & \sqsubseteq & \exists \mathit{loves}.\{\mathit{mary}\} \\
\mathit{Norwegian} & \sqsubseteq & \exists \mathit{comesFrom}.\{\mathit{norway}\} \\
\{\mathit{adam}\} & \sqsubseteq & \neg\{\mathit{eve}\} \\
\mathit{hasFather} \circ \mathit{hasBrother} & \sqsubseteq & \mathit{hasUncle}
\end{array}$$

$$\begin{array}{rcll}
\exists R.\top & \sqsubseteq & C & \text{Domain} \\
\top & \sqsubseteq & \forall R.C & \text{Range} \\
R \circ R & \sqsubseteq & R & \text{Transitivity} \\
\top & \sqsubseteq & \leq 1\, R.\top & \text{Functionality} \\
R & \sqsubseteq & R^{-} & \text{Symmetry} \\
R & \sqsubseteq & \neg R^{-} & \text{Asymmetry}
\end{array}$$

Complexity results

http://www.cs.man.ac.uk/~ezolin/dl/

Common restricted languages: $\mathcal{EL}$

The description logic $\mathcal{EL}$ allow the following concepts:

Common restricted languages: $\mathcal{EL}$

The description logic $\mathcal{EL}$ allow the following concepts:

$$
\begin{array}{rlcl}
C, D \rightarrow & A & \mid & \text{(atomic concept)} \\
& \top & \mid & \text{(universal concept)} \\
& \bot & \mid & \text{(bottom concept)} \\
& \{a\} & \mid & \text{(\textit{singular} enumeration)} \\
& C \sqcap D & \mid & \text{(intersection)} \\
& \exists R.C & \mid & \text{(existential restriction)}
\end{array}
$$

Common restricted languages: $\mathcal{EL}$

The description logic $\mathcal{EL}$ allow the following concepts:

$$
\begin{array}{rlcl}
C, D \to & A & \mid & \text{(atomic concept)} \\
& \top & \mid & \text{(universal concept)} \\
& \bot & \mid & \text{(bottom concept)} \\
& \{a\} & \mid & \text{(\textit{singular} enumeration)} \\
& C \sqcap D & \mid & \text{(intersection)} \\
& \exists R.C & \mid & \text{(existential restriction)}
\end{array}
$$

with the following axioms:

- $C \sqsubseteq D$ and $C \equiv D$ for concept descriptions D and C.
- $P \sqsubseteq Q$ and $P \equiv Q$ for roles P, Q.
- $C(a)$ and $R(a, b)$ for concept C, role R and individuals a, b.

Common restricted languages: $\mathcal{EL}$

Not supported (excerpt):

- negation, (only disjointness of classes: $C \sqcap D \sqsubseteq \bot$),
- disjunction,
- universal quantification,
- cardinalities,
- inverse roles,
- plus some role characteristics.

Common restricted languages: $\mathcal{EL}$

Not supported (excerpt):

- negation, (only disjointness of classes: $C \sqcap D \sqsubseteq \bot$),
- disjunction,
- universal quantification,
- cardinalities,
- inverse roles,
- plus some role characteristics.

- Captures language used for many large ontologies.
- Checking ontology consistency, class expression subsumption, and instance checking is in **P**.
- "Good for large ontologies."

Common restricted languages: *DL-Lite*

The description logic $\textit{DL-Lite}_R$ allows the following concepts:

Common restricted languages: *DL-Lite*

The description logic $DL\text{-}Lite_R$ allows the following concepts:

$$
\begin{array}{rlcl}
C \rightarrow & A & \mid & \text{(atomic concept)} \\
 & \exists R.\top & \mid & \text{(existential restriction with } \top \text{ only)} \\
D \rightarrow & A & \mid & \text{(atomic concept)} \\
 & \exists R.D & \mid & \text{(existential restriction)} \\
 & \neg D & \mid & \text{(negation)} \\
 & D \sqcap D' & \mid & \text{(intersection)}
\end{array}
$$

Common restricted languages: *DL-Lite*

The description logic $\textit{DL-Lite}_R$ allows the following concepts:

$$
\begin{array}{rlcl}
C \to & A & \mid & \text{(atomic concept)} \\
 & \exists R.\top & \mid & \text{(existential restriction with } \top \text{ only)} \\
D \to & A & \mid & \text{(atomic concept)} \\
 & \exists R.D & \mid & \text{(existential restriction)} \\
 & \neg D & \mid & \text{(negation)} \\
 & D \sqcap D' & \mid & \text{(intersection)}
\end{array}
$$

with the following axioms:

- $C \sqsubseteq D$ for concept descriptions D and C (and $C \equiv C'$).
- $P \sqsubseteq Q$ and $P \equiv Q$ for roles P, Q.
- $C(a)$ and $R(a, b)$ for concept C, role R and individuals a, b.

Common restricted languages: *DL-Lite*

Not supported (excerpt):

- disjunction,
- universal quantification,
- cardinalities,
- functional roles, keys,
- enumerations (closed classes),
- subproperties of chains, transitivity

Common restricted languages: *DL-Lite*

Not supported (excerpt):

- disjunction,
- universal quantification,
- cardinalities,
- functional roles, keys,
- enumerations (closed classes),
- subproperties of chains, transitivity

- Captures language for which queries can be translated to SQL.
  - Conjunctive queries over a *DL-Lite* knowledge base can be expanded with the TBox to a conjunctive query that can be answered over the Abox alone. This is called *first order rewritability*.
- "Good for large datasets."

Common restricted languages: $\mathcal{RL}$

The description logic $\mathcal{RL}$ (also called DLP) allow the following concepts:

Common restricted languages: $\mathcal{RL}$

The description logic $\mathcal{RL}$ (also called DLP) allow the following concepts:

$$
\begin{array}{rlll}
C \rightarrow & A & | & \text{(atomic concept)} \\
& C \sqcap C' & | & \text{(intersection)} \\
& C \sqcup C' & | & \text{(union)} \\
& \exists R.C & | & \text{(existential restriction)} \\
D \rightarrow & A & | & \text{(atomic concept)} \\
& D \sqcap D' & | & \text{(intersection)} \\
& \forall R.D & | & \text{(universal restriction)}
\end{array}
$$

Common restricted languages: $\mathcal{RL}$

The description logic $\mathcal{RL}$ (also called DLP) allow the following concepts:

$$
\begin{array}{rlcl}
C \rightarrow & A & \mid & \text{(atomic concept)} \\
& C \sqcap C' & \mid & \text{(intersection)} \\
& C \sqcup C' & \mid & \text{(union)} \\
& \exists R.C & \mid & \text{(existential restriction)} \\
D \rightarrow & A & \mid & \text{(atomic concept)} \\
& D \sqcap D' & \mid & \text{(intersection)} \\
& \forall R.D & \mid & \text{(universal restriction)}
\end{array}
$$

with the following axioms:

- $C \sqsubseteq D$, $C \equiv C'$, $\top \sqsubseteq \forall P.D$, $\top \sqsubseteq \forall P^{-}.D$, $P \sqsubseteq Q$, $P \equiv Q^{-}$ and $P \equiv Q$ for roles P, Q and concept descriptions D and C.
- $C(a)$ and $R(a, b)$ for concept C, role R and individuals a, b.

Contents

OWL and the Semantic Web

– OWL (The Web Ontology Language) is a set of ontology languages with semantics based on description logics.

OWL and the Semantic Web

- OWL (The Web Ontology Language) is a set of ontology languages with semantics based on description logics.
- They combine Web technology with description logic to make an *intelligent* web of data.

OWL and the Semantic Web

- OWL (The Web Ontology Language) is a set of ontology languages with semantics based on description logics.
- They combine Web technology with description logic to make an *intelligent* web of data.
- In OWL, all individuals, concepts and roles are assigned a URI (Unique Resource Identifier).

OWL and the Semantic Web

- OWL (The Web Ontology Language) is a set of ontology languages with semantics based on description logics.
- They combine Web technology with description logic to make an *intelligent* web of data.
- In OWL, all individuals, concepts and roles are assigned a URI (Unique Resource Identifier).
  - These URIs can be URLs, hence they can state where we can find more information about an item.

OWL and the Semantic Web

- OWL (The Web Ontology Language) is a set of ontology languages with semantics based on description logics.
- They combine Web technology with description logic to make an *intelligent* web of data.
- In OWL, all individuals, concepts and roles are assigned a URI (Unique Resource Identifier).
  - These URIs can be URLs, hence they can state where we can find more information about an item.
  - URIs can be set to be equal, so we can link two ontologies together by stating which URIs denote the same thing in different contexts.

OWL and the Semantic Web

- OWL (The Web Ontology Language) is a set of ontology languages with semantics based on description logics.
- They combine Web technology with description logic to make an *intelligent* web of data.
- In OWL, all individuals, concepts and roles are assigned a URI (Unique Resource Identifier).
  - These URIs can be URLs, hence they can state where we can find more information about an item.
  - URIs can be set to be equal, so we can link two ontologies together by stating which URIs denote the same thing in different contexts.
- OWL provides a concrete syntax for writing axioms, implementations of reasoners over the axioms, and a query language that applies the reasoners for knowledge extraction.

OWL 2 Profiles

– OWL has various *profiles* that correspond to different DLs.

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;
- OWL DL: corresponds to $\mathcal{SHION}(\mathcal{D})$;

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;
- OWL DL: corresponds to $\mathcal{SHION}(\mathcal{D})$;
- OWL 2 DL: corresponds to $\mathcal{SROIQ}(\mathcal{D})$ and is the "normal" OWL 2 (sublanguage): "maximum" expressivity while keeping reasoning problems decidable—but still very expensive;

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;
- OWL DL: corresponds to $\mathcal{SHION}(\mathcal{D})$;
- OWL 2 DL: corresponds to $\mathcal{SROIQ}(\mathcal{D})$ and is the "normal" OWL 2 (sublanguage): "maximum" expressivity while keeping reasoning problems decidable—but still very expensive;
- (Other) profiles are tailored for specific ends, e.g.,
  - OWL 2 QL: Corresponds to DL-Lite$_{\mathcal{R}}$, and is specifically designed for efficient database integration;

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;
- OWL DL: corresponds to $\mathcal{SHION}(\mathcal{D})$;
- OWL 2 DL: corresponds to $\mathcal{SROIQ}(\mathcal{D})$ and is the “normal” OWL 2 (sublanguage): “maximum” expressivity while keeping reasoning problems decidable—but still very expensive;
- (Other) profiles are tailored for specific ends, e.g.,
  - OWL 2 QL: Corresponds to DL-Lite$_{\mathcal{R}}$, and is specifically designed for efficient database integration;
  - OWL 2 EL: Corresponds to $\mathcal{EL}$, and is a lightweight language with polynomial time reasoning;

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;
- OWL DL: corresponds to $\mathcal{SHION}(\mathcal{D})$;
- OWL 2 DL: corresponds to $\mathcal{SROIQ}(\mathcal{D})$ and is the "normal" OWL 2 (sublanguage): "maximum" expressivity while keeping reasoning problems decidable—but still very expensive;
- (Other) profiles are tailored for specific ends, e.g.,
  - OWL 2 QL: Corresponds to DL-Lite$_{\mathcal{R}}$, and is specifically designed for efficient database integration;
  - OWL 2 EL: Corresponds to $\mathcal{EL}$, and is a lightweight language with polynomial time reasoning;
  - OWL 2 RL: Corresponds to $\mathcal{RL}$, and is designed for compatibility with rule-based inference tools.

OWL 2 Profiles

- OWL has various *profiles* that correspond to different DLs.
- OWL Lite: $\mathcal{SHIF}(\mathcal{D})$;
- OWL DL: corresponds to $\mathcal{SHION}(\mathcal{D})$;
- OWL 2 DL: corresponds to $\mathcal{SROIQ}(\mathcal{D})$ and is the “normal” OWL 2 (sublanguage): “maximum” expressivity while keeping reasoning problems decidable—but still very expensive;
- (Other) profiles are tailored for specific ends, e.g.,
  - OWL 2 QL: Corresponds to DL-Lite$_{\mathcal{R}}$, and is specifically designed for efficient database integration;
  - OWL 2 EL: Corresponds to $\mathcal{EL}$, and is a lightweight language with polynomial time reasoning;
  - OWL 2 RL: Corresponds to $\mathcal{RL}$, and is designed for compatibility with rule-based inference tools.
- OWL Full (not a proper DL): Anything goes: classes, relations, individuals, highly expressive, not decidable. But we want OWL’s reasoning capabilities, so stay away if you can—and you almost always can.

Contents

What cannot be expressed in DLs: Brothers

- Given terms

$$hasSibling \qquad Male$$

- . . . a brother is *defined* to be a sibling who is male
- In FOL: $\forall x \forall y \, (hasSibling(x, y) \wedge Male(y) \leftrightarrow hasBrother(x, y))$

What cannot be expressed in DLs: Brothers

- Given terms

$$hasSibling \qquad Male$$

- ...a brother is *defined* to be a sibling who is male
- In FOL: $\forall x \forall y\, (hasSibling(x, y) \wedge Male(y) \leftrightarrow hasBrother(x, y))$
- Best try:

$$\begin{array}{l} hasBrother \sqsubseteq hasSibling \\ \top \sqsubseteq \forall hasBrother.Male \\ \exists hasSibling.Male \sqsubseteq \exists hasBrother.\top \end{array}$$

- Not enough to infer that *all* male siblings are brothers

What cannot be expressed in DLs: Diamond Properties

- A semi-detached house has a left and a right unit
- Each unit has a separating wall
- The separating walls of the left and right units are the same

What cannot be expressed in DLs: Diamond Properties

- A semi-detached house has a left and a right unit
- Each unit has a separating wall
- The separating walls of the left and right units are the same
- In FOL:

$$\begin{aligned}&\forall x(\mathit{SemiDetached}(x) \leftrightarrow \\ &\quad \exists y \exists z(\mathit{hasLeftUnit}(x, y) \wedge \mathit{Unit}(y) \wedge \mathit{hasRightUnit}(x, z) \wedge \mathit{Unit}(z) \wedge \\ &\qquad \exists w(\mathit{Wall}(w) \wedge \mathit{hasSeparatingWall}(x, w) \wedge \mathit{hasSeparatingWall}(y, w)))\end{aligned}$$

What cannot be expressed in DLs: Diamond Properties

- A semi-detached house has a left and a right unit
- Each unit has a separating wall
- The separating walls of the left and right units are the same
- In FOL:

$$\forall x(\mathit{SemiDetached}(x) \leftrightarrow \\ \exists y \exists z(\mathit{hasLeftUnit}(x, y) \wedge \mathit{Unit}(y) \wedge \mathit{hasRightUnit}(x, z) \wedge \mathit{Unit}(z) \wedge \\ \exists w(\mathit{Wall}(w) \wedge \mathit{hasSeparatingWall}(x, w) \wedge \mathit{hasSeparatingWall}(y, w)))$$

- "diamond property"

What cannot be expressed in DLs: Diamond Properties

- A semi-detached house has a left and a right unit
- Each unit has a separating wall
- The separating walls of the left and right units are the same
- In FOL:

$$\begin{aligned}
&\forall x(\mathit{SemiDetached}(x) \leftrightarrow \\
&\quad \exists y \exists z(\mathit{hasLeftUnit}(x, y) \wedge \mathit{Unit}(y) \wedge \mathit{hasRightUnit}(x, z) \wedge \mathit{Unit}(z) \wedge \\
&\qquad \exists w(\mathit{Wall}(w) \wedge \mathit{hasSeparatingWall}(x, w) \wedge \mathit{hasSeparatingWall}(y, w)))
\end{aligned}$$

- "diamond property"
- Try

$$\begin{aligned}
\mathit{SemiDetached} \equiv\ & \exists \mathit{hasLeftUnit}.(\mathit{Unit} \sqcap \exists \mathit{hasSeparatingWall}.\mathit{Wall}) \sqcap \\
& \exists \mathit{hasRightUnit}.(\mathit{Unit} \sqcap \exists \mathit{hasSeparatingWall}.\mathit{Wall})
\end{aligned}$$

- No way of stating that the walls are the same.

What cannot be expressed in DLs: Connecting Properties

– Given terms

$$\mathit{Person} \qquad \mathit{hasChild} \qquad \mathit{hasBirthday}$$

– A twin parent is defined to be a person who has two children with the same birthday.

What cannot be expressed in DLs: Connecting Properties

- Given terms

$$\mathit{Person} \qquad \mathit{hasChild} \qquad \mathit{hasBirthday}$$

- A twin parent is defined to be a person who has two children with the same birthday.
- Try...

$$\begin{aligned} \mathit{TwinParent} \equiv \mathit{Person} \quad & \sqcap\ \exists \mathit{hasChild}.\exists \mathit{hasBirthday}[\ldots] \\ & \sqcap\ \exists \mathit{hasChild}.\exists \mathit{hasBirthday}[\ldots] \end{aligned}$$

- No way to connect the two birthdays to say that they're the same.
  - (and no way to say that the children are *not* the same)

What cannot be expressed in DLs: Connecting Properties

– Given terms

$$Person \qquad hasChild \qquad hasBirthday$$

– A twin parent is defined to be a person who has two children with the same birthday.

– Try...

$$\begin{aligned} TwinParent \equiv Person \quad & \sqcap\ \exists hasChild.\exists hasBirthday[\ldots] \\ & \sqcap\ \exists hasChild.\exists hasBirthday[\ldots] \end{aligned}$$

– No way to connect the two birthdays to say that they're the same.

  – (and no way to say that the children are *not* the same)

– Try...

$$TwinParent \equiv Person \sqcap \geq_2 hasChild.\exists hasBirthday[\ldots]$$

– Still no way of connecting the birthdays

Reasoning about Numbers

- Reasoning about natural numbers is undecidable in general.
- DL Reasoning is decidable
- Therefore, general reasoning about numbers can't be "encoded" in DL

Reasoning about Numbers

- Reasoning about natural numbers is undecidable in general.
- DL Reasoning is decidable
- Therefore, general reasoning about numbers can't be "encoded" in DL
- For instance, *there is no largest prime number*:

$$\forall n.\exists p.(p > n \wedge \forall k, l.p = k \cdot l \rightarrow (k = 1 \vee l = 1))$$

- Could try...

$$\begin{gathered}
Number(zero) \\
Number \sqsubseteq \exists hasSuccessor.Number \\
\top \sqsubseteq \leq 1\, hasSuccessor.\top \\
hasSuccessor \sqsubseteq lessThan \\
lessThan \circ lessThan \sqsubseteq lessThan \\
lessThan \sqsubseteq \neg lessThan^{-}
\end{gathered}$$

- Cannot encode addition, multiplication, etc.
- Note: a lot can be done with other logics, but not with DLs

Reasoning about Numbers

- Reasoning about natural numbers is undecidable in general.
- DL Reasoning is decidable
- Therefore, general reasoning about numbers can't be "encoded" in DL
- For instance, *there is no largest prime number*:

$$\forall n.\exists p.(p > n \wedge \forall k, l.p = k \cdot l \to (k = 1 \vee l = 1))$$

- Could try...

$$\begin{gathered} Number(zero) \\ Number \sqsubseteq \exists hasSuccessor.Number \\ \top \sqsubseteq \leq 1\, hasSuccessor.\top \\ hasSuccessor \sqsubseteq lessThan \\ lessThan \circ lessThan \sqsubseteq lessThan \\ lessThan \sqsubseteq \neg lessThan^{-} \end{gathered}$$

- Cannot encode addition, multiplication, etc.
- Note: a lot can be done with other logics, but not with DLs
  - Outside the intended scope of Description Logics

Contents

FO-rewritability

Assume $\mathcal{T}_{\mathcal{L}}$ is the set of TBoxes over the language $\mathcal{L}$, and that UCQ is the set of queries that are unions of conjunctive queries, and let

$$\mathcal{K} \models q_1 \vee q_2 \Leftrightarrow \mathcal{K} \models q_1 \text{ or } \mathcal{K} \models q_2$$
$$\mathcal{K} \models q_1 \wedge q_2 \Leftrightarrow \mathcal{K} \models q_1 \text{ and } \mathcal{K} \models q_2$$

A description logic $\mathcal{L}$ enjoys *first order rewritability* if there exists a rewriting function $\rho : \mathcal{T}_{\mathcal{L}} \times UCQ \to UCQ$, such that for any knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ over $\mathcal{L}$ and any conjunctive query $q(\vec{x})$ over $\mathcal{K}$ we have that

$$\mathcal{A} \models \rho(\mathcal{T}, q(\vec{a})) \Leftrightarrow \mathcal{K} \models q(\vec{a})$$

This allows us to divide the querying up into two stages: i) translation of the query, and ii) ABox querying. This is useful for e.g. translating a query from a DL query to an SQL query where the ABox is a relational database.

E.g. let $\mathcal{T} := \{C_1 \sqsubseteq D, C_2 \sqsubseteq D, A \sqsubseteq C_1\}$ and $q(x) := D(x)$ we have that for any Abox $\mathcal{A}$ that

$$\mathcal{A} \models D(a) \vee C_1(a) \vee C_2(a) \vee A(a) \Leftrightarrow \langle \mathcal{T}, \mathcal{A} \rangle \models D(a)$$