INF3580/4580 – Semantic Technologies – Spring 2017

Lecture 11: OWL 2

Leif Harald Karlsen

27th March 2017

Department of Informatics

University of Oslo

Outline

1. Reminder: $\mathcal{ALC}$
2. Important assumptions
3. OWL 2
   - Axioms and assertions using individuals
   - Restrictions on roles
   - Modelling problems
   - Roles
   - Datatypes

The $\mathcal{ALC}$ Description Logic

Vocabulary

Fix a set of *atomic concepts* $\{A_1, A_2, \ldots\}$, *roles* $\{R_1, R_2, \ldots\}$ and individuals $\{a_1, a_2, \ldots\}$.

The $\mathcal{ALC}$ Description Logic

Vocabulary

Fix a set of *atomic concepts* $\{A_1, A_2, \ldots\}$, *roles* $\{R_1, R_2, \ldots\}$ and individuals $\{a_1, a_2, \ldots\}$.

$\mathcal{ALC}$ concept descriptions

$$
\begin{array}{rlll}
C, D \rightarrow & A_i & | & \text{(atomic concept)} \\
& \top & | & \text{(universal concept)} \\
& \bot & | & \text{(bottom concept)}
\end{array}
$$

The $\mathcal{ALC}$ Description Logic

Vocabulary

Fix a set of *atomic concepts* $\{A_1, A_2, \ldots\}$, *roles* $\{R_1, R_2, \ldots\}$ and individuals $\{a_1, a_2, \ldots\}$.

$\mathcal{ALC}$ concept descriptions

$$
\begin{array}{rlcl}
C, D \rightarrow & A_i & | & \text{(atomic concept)} \\
& \top & | & \text{(universal concept)} \\
& \bot & | & \text{(bottom concept)} \\
& \neg C & | & \text{(negation)} \\
& C \sqcap D & | & \text{(intersection)} \\
& C \sqcup D & | & \text{(union)}
\end{array}
$$

The $\mathcal{ALC}$ Description Logic

Vocabulary

Fix a set of *atomic concepts* $\{A_1, A_2, \ldots\}$, *roles* $\{R_1, R_2, \ldots\}$ and individuals $\{a_1, a_2, \ldots\}$.

$\mathcal{ALC}$ concept descriptions

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

The $\mathcal{ALC}$ Description Logic

Vocabulary

Fix a set of *atomic concepts* $\{A_1, A_2, \dots\}$, *roles* $\{R_1, R_2, \dots\}$ and individuals $\{a_1, a_2, \dots\}$.

$\mathcal{ALC}$ concept descriptions

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

Axioms

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

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each atomic concept A, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role R, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual a.

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each atomic concept A, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role R, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual a.

Interpretation of concept descriptions

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ \bot^{\mathcal{I}} &= \emptyset \end{aligned}$$

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each atomic concept A, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role R, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual a.

Interpretation of concept descriptions

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}
\end{aligned}$$

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each atomic concept A, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role R, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual a.

Interpretation of concept descriptions

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{for all } b, \text{ if } \langle a, b\rangle \in R^{\mathcal{I}} \text{ then } b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{there is a } b \text{ where } \langle a, b\rangle \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}
\end{aligned}
$$

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each atomic concept A, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role R, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual a.

Interpretation of concept descriptions

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{for all } b \text{, if } \langle a, b\rangle \in R^{\mathcal{I}} \text{ then } b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{there is a } b \text{ where } \langle a, b\rangle \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}
\end{aligned}
$$

Interpretation of Axioms

- $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $\mathcal{I} \models C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$
- $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I} \models R(a, b)$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$.

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$

$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$

$$Penguin(a) \qquad eats(a, b)$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$\begin{aligned} &\mathit{Penguin} \sqsubseteq \mathit{Animal} \sqcap \forall \mathit{eats}.\mathit{Fish} & \mathit{Fish} \sqsubseteq \mathit{Animal} \\ &\mathit{Penguin} \sqcap \mathit{Fish} \sqsubseteq \bot & \mathit{Animal} \sqsubseteq \exists \mathit{eats}.\top \\ &\mathit{Penguin}(a) & \mathit{eats}(a, b) \end{aligned}$$

Let $\mathcal{I}$ be an interpretation such that

$$\begin{aligned} &\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{\mathit{tweety}, \mathit{terry}, \mathit{carl}\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = \mathit{tweety}, \quad b^{\mathcal{I}} = \mathit{terry} \\ &\mathit{Penguin}^{\mathcal{I}} = \{a^{\mathcal{I}} \\ &\mathit{eats}^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle, \end{aligned}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}$$
$$eats^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle,$$
$$Fish^{\mathcal{I}} = \{b^{\mathcal{I}}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a,b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}$$
$$eats^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle,$$
$$Fish^{\mathcal{I}} = \{b^{\mathcal{I}}$$
$$Animal^{\mathcal{I}} = \{a^{\mathcal{I}}, b^{\mathcal{I}}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}$$
$$eats^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle, \langle b^{\mathcal{I}}, carl\rangle\}$$
$$Fish^{\mathcal{I}} = \{b^{\mathcal{I}}$$
$$Animal^{\mathcal{I}} = \{a^{\mathcal{I}}, b^{\mathcal{I}}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\begin{aligned}
&\Delta^\mathcal{I} = \top^\mathcal{I} = \{tweety, terry, carl\}, \quad \bot^\mathcal{I} = \emptyset, \quad a^\mathcal{I} = tweety, \quad b^\mathcal{I} = terry\\
&Penguin^\mathcal{I} = \{a^\mathcal{I}\\
&eats^\mathcal{I} = \{\langle a^\mathcal{I}, b^\mathcal{I}\rangle, \langle b^\mathcal{I}, carl\rangle\} = \{\langle tweety, terry\rangle, \langle terry, carl\rangle\}\\
&Fish^\mathcal{I} = \{b^\mathcal{I}\\
&Animal^\mathcal{I} = \{a^\mathcal{I}, b^\mathcal{I}
\end{aligned}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}\} = \{tweety\}$$
$$eats^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle, \langle b^{\mathcal{I}}, carl\rangle\} = \{\langle tweety, terry\rangle, \langle terry, carl\rangle\}$$
$$Fish^{\mathcal{I}} = \{b^{\mathcal{I}}\} = \{terry\}$$
$$Animal^{\mathcal{I}} = \{a^{\mathcal{I}}, b^{\mathcal{I}}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}\} = \{tweety\}$$
$$eats^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle, \langle b^{\mathcal{I}}, carl\rangle\} = \{\langle tweety, terry\rangle, \langle terry, carl\rangle\}$$
$$Fish^{\mathcal{I}} = \{b^{\mathcal{I}}\} = \{terry\}$$
$$Animal^{\mathcal{I}} = \{a^{\mathcal{I}}, b^{\mathcal{I}}\} = \{tweety, terry\}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{I}$ be an interpretation such that

$$\Delta^{\mathcal{I}} = \top^{\mathcal{I}} = \{tweety, terry, carl\}, \quad \bot^{\mathcal{I}} = \emptyset, \quad a^{\mathcal{I}} = tweety, \quad b^{\mathcal{I}} = terry$$
$$Penguin^{\mathcal{I}} = \{a^{\mathcal{I}}\} = \{tweety\}$$
$$eats^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle, \langle b^{\mathcal{I}}, carl\rangle\} = \{\langle tweety, terry\rangle, \langle terry, carl\rangle\}$$
$$Fish^{\mathcal{I}} = \{b^{\mathcal{I}}\} = \{terry\}$$
$$Animal^{\mathcal{I}} = \{a^{\mathcal{I}}, b^{\mathcal{I}}\} = \{tweety, terry\}$$

Now $\mathcal{I} \models \mathcal{K}$.

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$

$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$

$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{J}$ be an interpretation such that

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{J}$ be an interpretation such that

$$\Delta^{\mathcal{J}} = \top^{\mathcal{J}} = \{tweety\}, \quad \bot^{\mathcal{J}} = \emptyset, \quad a^{\mathcal{J}} = tweety, b^{\mathcal{J}} = tweety$$
$$Animal^{\mathcal{J}} = \{a^{\mathcal{J}}, b^{\mathcal{J}}\} = \{tweety\},$$
$$Penguin^{\mathcal{J}} = \{a^{\mathcal{J}}\} = \{tweety\},$$
$$Fish^{\mathcal{J}} = \{b^{\mathcal{J}}\} = \{tweety\}$$
$$eats^{\mathcal{J}} = \{\langle a^{\mathcal{J}}, b^{\mathcal{J}}\rangle, \langle b^{\mathcal{J}}, a^{\mathcal{J}}\rangle\} = \{\langle tweety, tweety\rangle\}$$

$\mathcal{ALC}$ Examples

Let $\mathcal{K}$ be the following set of axioms:

$$Penguin \sqsubseteq Animal \sqcap \forall eats.Fish \qquad Fish \sqsubseteq Animal$$
$$Penguin \sqcap Fish \sqsubseteq \bot \qquad Animal \sqsubseteq \exists eats.\top$$
$$Penguin(a) \qquad eats(a, b)$$

Let $\mathcal{J}$ be an interpretation such that

$$\Delta^{\mathcal{J}} = \top^{\mathcal{J}} = \{tweety\}, \quad \bot^{\mathcal{J}} = \emptyset, \quad a^{\mathcal{J}} = tweety, b^{\mathcal{J}} = tweety$$
$$Animal^{\mathcal{J}} = \{a^{\mathcal{J}}, b^{\mathcal{J}}\} = \{tweety\},$$
$$Penguin^{\mathcal{J}} = \{a^{\mathcal{J}}\} = \{tweety\},$$
$$Fish^{\mathcal{J}} = \{b^{\mathcal{J}}\} = \{tweety\}$$
$$eats^{\mathcal{J}} = \{\langle a^{\mathcal{J}}, b^{\mathcal{J}}\rangle, \langle b^{\mathcal{J}}, a^{\mathcal{J}}\rangle\} = \{\langle tweety, tweety\rangle\}$$

Now $\mathcal{J} \not\models \mathcal{K}$ since $\mathcal{J} \not\models Penguin \sqcap Fish \sqsubseteq \bot$.

Modelling patterns

So, what can we say with $\mathcal{ALC}$?

- ✓ Every person has a mother.
- ✓ Penguins eats only fish. Horses eats only chocolate.
- ✗ Every nuclear family has two parents, at least two children and a dog.
- ✓ No smoker is a non-smoker (and vice versa).
- ✗ Everybody loves Mary.
- ✗ Adam is not Eve (and vice versa).
- ✓ Everything is black or white.
- ✓ There is no such thing as a free lunch.
- ✗ Brothers of fathers are uncles.
- ✗ My friend's friends are also my friends.
- ✗ If Homer is married to Marge, then Marge is married to Homer.
- ✗ If Homer is a parent of Bart, then Bart is a child of Homer.

Today we'll learn how to say more.

Outline

1. Reminder: $\mathcal{ALC}$
2. Important assumptions
3. OWL 2
   - Axioms and assertions using individuals
   - Restrictions on roles
   - Modelling problems
   - Roles
   - Datatypes

World assumptions

- Closed World Assumption (CWA)
- Open World Assumption (OWA)

CWA:

- Complete knowledge.
- Any statement that is not known to be true is false. $(*)$
- Typical semantics for database systems.

OWA:

- Potential incomplete knowledge.
- $(*)$ does not hold.
- Typical semantics for logic-based systems.

Name assumptions

- Unique name assumption (UNA)
- Non-unique name assumption (NUNA)

- Under any assumption, equal names (read: individual URIs, DB constants) always denote the same "thing" (obviously).
  - E.g., cannot have $a^{\mathcal{I}} \neq a^{\mathcal{I}}$.
- Under UNA, different names *always* denote different things.
  - E.g., $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.
  - common in relational databases.
- Under NUNA, different names *need not* denote different things.
  - Can have , $a^{\mathcal{I}} = b^{\mathcal{I}}$, or
  - $\texttt{dbpedia:Oslo}^{\mathcal{I}} = \texttt{geo:34521}^{\mathcal{I}}$.

Outline

1. Reminder: $\mathcal{ALC}$
2. Important assumptions
3. OWL 2
   - Axioms and assertions using individuals
   - Restrictions on roles
   - Modelling problems
   - Roles
   - Datatypes

$\mathcal{SHOIN}(\mathcal{D})$ and OWL 2

- OWL 2 is based on the DL $\mathcal{SHOIN}(\mathcal{D})$:
  - $\mathcal{S}$ for $\mathcal{ALC}$[1] plus role transitivity,
  - $\mathcal{H}$ for roles hierarchies,
  - $\mathcal{O}$ for closed classes,
  - $\mathcal{I}$ for inverse roles,
  - $\mathcal{N}$ for cardinality restrictions, and
  - $\mathcal{D}$ for datatypes.

[1] Attributive Concept Language with Complements

$\mathcal{SHOIN}(\mathcal{D})$ and OWL 2

- OWL 2 is based on the DL $\mathcal{SHOIN}(\mathcal{D})$:
  - $\mathcal{S}$ for $\mathcal{ALC}$[1] plus role transitivity,
  - $\mathcal{H}$ for roles hierarchies,
  - $\mathcal{O}$ for closed classes,
  - $\mathcal{I}$ for inverse roles,
  - $\mathcal{N}$ for cardinality restrictions, and
  - $\mathcal{D}$ for datatypes.
- So, today we'll see:
  - new concept and role builders,
  - new TBox axioms,
  - new ABox axioms,
  - new RBox axioms, and
  - datatypes.

[1] Attributive Concept Language with Complements

Individual identity

- New ABox axioms.
- Express equality and non-equality between individuals.

Individual identity

- New ABox axioms.
- Express equality and non-equality between individuals.
- Syntax:
  - DL: $a = b$, $a \neq b$;
  - RDF/OWL: `:a owl:sameAs :b`, `:a owl:differentFrom :b`,
  - Manchester: `SameAs`, `DifferentFrom`.

Individual identity

- New ABox axioms.
- Express equality and non-equality between individuals.
- Syntax:
  - DL: $a = b$, $a \neq b$;
  - RDF/OWL: `:a owl:sameAs :b`, `:a owl:differentFrom :b`,
  - Manchester: `SameAs`, `DifferentFrom`.
- Semantics:
  - $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$
  - $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$

Individual identity

- New ABox axioms.
- Express equality and non-equality between individuals.
- Syntax:
  - DL: $a = b$, $a \neq b$;
  - RDF/OWL: `:a owl:sameAs :b`, `:a owl:differentFrom :b`,
  - Manchester: `SameAs`, `DifferentFrom`.
- Semantics:
  - $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$
  - $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$
- Examples:
  - `sim:Bart owl:sameAs dbpedia:Bart_Simpson`,
  - `sim:Bart owl:differentFrom sim:Homer`.

Individual identity

- New ABox axioms.
- Express equality and non-equality between individuals.
- Syntax:
  - DL: $a = b$, $a \neq b$;
  - RDF/OWL: `:a owl:sameAs :b`, `:a owl:differentFrom :b`,
  - Manchester: `SameAs`, `DifferentFrom`.
- Semantics:
  - $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$
  - $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$
- Examples:
  - `sim:Bart owl:sameAs dbpedia:Bart_Simpson`,
  - `sim:Bart owl:differentFrom sim:Homer`.
- Remember:
  - Non unique name assumption (NUNA) in Sem. Web,
  - must sometimes use $=$ and $\neq$ to get expected results.

Creating concepts using individuals

- New concept builder.

Creating concepts using individuals

- New concept builder.
- Create (anonymous) concepts by explicitly listing all members.

Creating concepts using individuals

- New concept builder.
- Create (anonymous) concepts by explicitly listing all members.
- Called *closed classes* in OWL.

Creating concepts using individuals

- New concept builder.
- Create (anonymous) concepts by explicitly listing all members.
- Called *closed classes* in OWL.
- Syntax:
  - DL: $\{a, b, \ldots\}$
  - RDF/OWL: `owl:oneOf` + `rdf:List`++
  - Manchester: `{a, b, ...}`

Creating concepts using individuals

- New concept builder.
- Create (anonymous) concepts by explicitly listing all members.
- Called *closed classes* in OWL.
- Syntax:
  - DL: $\{a, b, \ldots\}$
  - RDF/OWL: `owl:oneOf` + `rdf:List`++
  - Manchester: `{a, b, ...}`
- Example:
  - $SimpsonFamily \equiv \{Homer, Marge, Bart, Lisa, Maggie\}$
  - `:SimpsonFamily owl:equivalentClass [owl:oneOf (:Homer :Marge :Bart :Lisa :Maggie)] .`

Creating concepts using individuals

- New concept builder.
- Create (anonymous) concepts by explicitly listing all members.
- Called *closed classes* in OWL.
- Syntax:
  - DL: $\{a, b, \ldots\}$
  - RDF/OWL: `owl:oneOf` + `rdf:List`++
  - Manchester: `{a, b, ...}`
- Example:
  - $SimpsonFamily \equiv \{Homer, Marge, Bart, Lisa, Maggie\}$
  - `:SimpsonFamily owl:equivalentClass [owl:oneOf (:Homer :Marge :Bart :Lisa :Maggie)] .`
- Note:
  - The individuals does not necessarily represent different objects,
  - we still need $=$ and $\neq$ to say that members are the same/different.
  - "Closed classes of data values" are *datatypes*.

Axioms involving individuals: Negative Property Assertions

- New ABox axiom.
- Syntax:
  - DL: $\neg R(a, b)$,
  - RDF/OWL: `owl:NegativePropertyAssertion` (Class of assertions/triples)
  - Manchester: `a not R b`.

Axioms involving individuals: Negative Property Assertions

- New ABox axiom.
- Syntax:
  - DL: $\neg R(a, b)$,
  - RDF/OWL: `owl:NegativePropertyAssertion` (Class of assertions/triples)
  - Manchester: `a not R b`.
- Semantics:
  - $\mathcal{I} \models \neg R(a, b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \notin R^{\mathcal{I}}$,

Axioms involving individuals: Negative Property Assertions

- New ABox axiom.
- Syntax:
  - DL: $\neg R(a, b)$,
  - RDF/OWL: `owl:NegativePropertyAssertion` (Class of assertions/triples)
  - Manchester: `a not R b`.
- Semantics:
  - $\mathcal{I} \models \neg R(a, b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \notin R^{\mathcal{I}}$,
- Notes:
  - Works both for object properties and datatype properties.

Axioms involving individuals: Negative Property Assertions

- New ABox axiom.
- Syntax:
  - DL: $\neg R(a, b)$,
  - RDF/OWL: `owl:NegativePropertyAssertion` (Class of assertions/triples)
  - Manchester: `a not R b`.
- Semantics:
  - $\mathcal{I} \models \neg R(a, b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \notin R^{\mathcal{I}}$,
- Notes:
  - Works both for object properties and datatype properties.
- Examples:
  - `:Bart not :hasFather :NedFlanders`
  - `:Bart not :hasAge "2"^^xsd:int`

Recap of existential and universal restrictions

- Existential restrictions
  - have the form $\exists R.D$,
  - typically used to connect classes,
  - $C \sqsubseteq \exists R.D$: A C is R-related to (at least) *some* D:
    - Example: A person has a female parent: $Person \sqsubseteq \exists hasParent.Woman$.
  - Note that C-objects can be R-related to other things:
    - A person may have other parents who are not women—but there must be one who's a woman.

Recap of existential and universal restrictions

- Existential restrictions
  - have the form $\exists R.D$,
  - typically used to connect classes,
  - $C \sqsubseteq \exists R.D$: A C is R-related to (at least) *some* D:
    - Example: A person has a female parent: $Person \sqsubseteq \exists hasParent.Woman$.
  - Note that C-objects can be R-related to other things:
    - A person may have other parents who are not women—but there must be one who's a woman.
- Universal restrictions
  - have the form $\forall R.D$,
  - restrict the things a type of object can be connected to,
  - $C \sqsubseteq \forall R.D$: C is R-related to D's *only*:
    - Example: A horse eats only chocolate: $Horse \sqsubseteq \forall eats.Chocolate$.
  - Note that C-objects may not be R-related to anything at all:
    - A horse does not have to eat anything—but if it does it must be chocolate.

Cardinality restrictions

- New concept builder.
- Syntax:
  - DL: $\leq_n R.D$ and $\geq_n R.D$ (and $=_n R.D$).
  - RDF/OWL: `owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`.
  - Manchester: `min`, `max`, `exactly`.

Cardinality restrictions

- New concept builder.
- Syntax:
  - DL: $\leq_n R.D$ and $\geq_n R.D$ (and $=_n R.D$).
  - RDF/OWL: `owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`.
  - Manchester: `min`, `max`, `exactly`.
- Semantics:
  - $(\leq_n R.D)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b : \langle a, b\rangle \in R^{\mathcal{I}} \wedge b \in D^{\mathcal{I}}\}| \leq n\}$
  - $(\geq_n R.D)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b : \langle a, b\rangle \in R^{\mathcal{I}} \wedge b \in D^{\mathcal{I}}\}| \geq n\}$
- Restricts the number of relations a type of object can/must have.

Cardinality restrictions

- New concept builder.
- Syntax:
  - DL: $\leq_n R.D$ and $\geq_n R.D$ (and $=_n R.D$).
  - RDF/OWL: `owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`.
  - Manchester: `min`, `max`, `exactly`.
- Semantics:
  - $(\leq_n R.D)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b : \langle a, b\rangle \in R^{\mathcal{I}} \wedge b \in D^{\mathcal{I}}\}| \leq n\}$
  - $(\geq_n R.D)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b : \langle a, b\rangle \in R^{\mathcal{I}} \wedge b \in D^{\mathcal{I}}\}| \geq n\}$
- Restricts the number of relations a type of object can/must have.
- TBox axioms read:
  - $C \sqsubseteq \Box_n R.D$: "A C is R-related to n number of D's."
    - $\leq$: *at most*
    - $\geq$: *at least*
    - $=$: *exactly*

Example cardinality restriction

- $Car \sqsubseteq \leq_2 driveAxle.\top$
  - "A car has at most two drive axles."

Example cardinality restriction

- $Car \sqsubseteq \leq_2 driveAxle.\top$
  - "A car has at most two drive axles."
- $RangeRover \sqsubseteq =_1 driveAxle.FrontAxle \sqcap =_1 driveAxle.RearAxle$
  - "A Range Rover has one front axle as drive axle and one rear axle as drive axle".

Example cardinality restriction

- $Car \sqsubseteq\ \leq_2 driveAxle.\top$
  - "A car has at most two drive axles."
- $RangeRover \sqsubseteq\ =_1 driveAxle.FrontAxle \sqcap =_1 driveAxle.RearAxle$
  - "A Range Rover has one front axle as drive axle and one rear axle as drive axle".
- $Human \sqsubseteq\ =_2 hasBiologicalParent.\top$
  - "A human has two biological parents."

Example cardinality restriction

- $\mathit{Car} \sqsubseteq \leq_2 \mathit{driveAxle}.\top$
  - "A car has at most two drive axles."
- $\mathit{RangeRover} \sqsubseteq =_1 \mathit{driveAxle}.\mathit{FrontAxle} \sqcap =_1 \mathit{driveAxle}.\mathit{RearAxle}$
  - "A Range Rover has one front axle as drive axle and one rear axle as drive axle".
- $\mathit{Human} \sqsubseteq =_2 \mathit{hasBiologicalParent}.\top$
  - "A human has two biological parents."
- $\mathit{Mammal} \sqsubseteq =_1 \mathit{hasParent}.\mathit{Female} \sqcap =_1 \mathit{hasParent}.\mathit{Male}$
  - "A mammal has one parent that is a female and one parent that is a male."

Example cardinality restriction

- $Car \sqsubseteq \leq_2 driveAxle.\top$
  - "A car has at most two drive axles."
- $RangeRover \sqsubseteq =_1 driveAxle.FrontAxle \sqcap =_1 driveAxle.RearAxle$
  - "A Range Rover has one front axle as drive axle and one rear axle as drive axle".
- $Human \sqsubseteq =_2 hasBiologicalParent.\top$
  - "A human has two biological parents."
- $Mammal \sqsubseteq =_1 hasParent.Female \sqcap =_1 hasParent.Male$
  - "A mammal has one parent that is a female and one parent that is a male."
- $\geq_2 owns.Houses \sqcup \geq_5 own.Car \sqsubseteq Rich$
  - "Everyone who owns more than two houses or five cars is rich."

One more value restriction

- Restrictions of the form $\forall R.D$, $\exists R.D$, $\leq_n R.D$, $\geq_n R.D$ are called *qualified* when D is not $\top$.
- We can also qualify with a closed class.

One more value restriction

- Restrictions of the form $\forall R.D$, $\exists R.D$, $\leq_n R.D$, $\geq_n R.D$ are called *qualified* when D is not $\top$.
- We can also qualify with a closed class.
- Syntax:
  - RDF/OWL: `hasValue`,
  - DL, Manchester: just use: $\{\ldots\}$.

One more value restriction

- Restrictions of the form $\forall R.D$, $\exists R.D$, $\leq_n R.D$, $\geq_n R.D$ are called *qualified* when D is not $\top$.
- We can also qualify with a closed class.
- Syntax:
  - RDF/OWL: `hasValue`,
  - DL, Manchester: just use: $\{\ldots\}$.
- Example:
  - $Bieberette \equiv Girl \sqcap \exists loves.\{J.Bieber\}$
  - $\top \sqsubseteq \exists loves.\{Mary\}$
  - $Norwegian \equiv Person \sqcap \exists citizenOf.\{Norway\}$

Self restriction

- New construct builder.
- Local reflexivity restriction. Restricts to objects which are related to themselves.
- Syntax:
  - DL: $\exists R.Self$
  - RDF/OWL: `owl:hasSelf`,
  - Manchester: `Self`

Self restriction

- New construct builder.
- Local reflexivity restriction. Restricts to objects which are related to themselves.
- Syntax:
  - DL: $\exists R.Self$
  - RDF/OWL: `owl:hasSelf`,
  - Manchester: `Self`
- Semantics:
  - $(\exists R.Self)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \langle x, x \rangle \in R^{\mathcal{I}}\}$

Self restriction

- New construct builder.
- Local reflexivity restriction. Restricts to objects which are related to themselves.
- Syntax:
  - DL: $\exists R.\mathit{Self}$
  - RDF/OWL: `owl:hasSelf`,
  - Manchester: `Self`
- Semantics:
  - $(\exists R.\mathit{Self})^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \langle x, x \rangle \in R^{\mathcal{I}}\}$
- Examples:
  - $\mathit{AutoregulatingProcess} \sqsubseteq \exists \mathit{regulate}.\mathit{Self}$
  - $\exists \mathit{hasBoss}.\mathit{Self} \sqsubseteq \mathit{SelfEmployed}$

Restrictions, non-unique names and open worlds

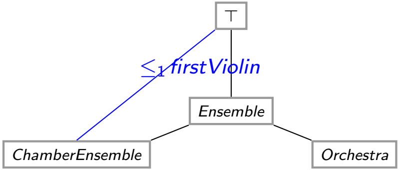
Restrictions + the OWA and the NUNA can be tricky, consider:

TBox:

$$Orchestra \sqsubseteq Ensemble$$

$$ChamberEnsemble \sqsubseteq Ensemble$$

$$ChamberEnsemble \sqsubseteq \leq_1 firstViolin.\top$$



Restrictions, non-unique names and open worlds

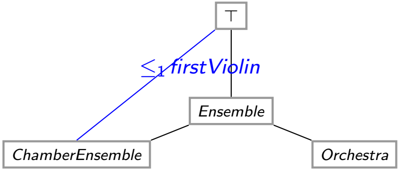
Restrictions + the OWA and the NUNA can be tricky, consider:

TBox:

$$Orchestra \sqsubseteq Ensemble$$
$$ChamberEnsemble \sqsubseteq Ensemble$$
$$ChamberEnsemble \sqsubseteq \leq_1 firstViolin.\top$$

ABox:

```
Ensemble(oslo)
firstViolin(oslo, skolem)
firstViolin(oslo, lie)
```



Restrictions, non-unique names and open worlds

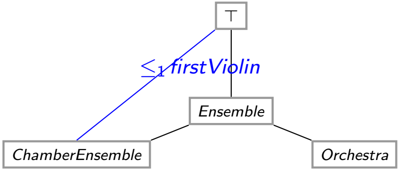
Restrictions + the OWA and the NUNA can be tricky, consider:

TBox:

$$Orchestra \sqsubseteq Ensemble$$
$$ChamberEnsemble \sqsubseteq Ensemble$$
$$ChamberEnsemble \sqsubseteq \leq_1 firstViolin.\top$$

ABox:

```
Ensemble(oslo)
firstViolin(oslo, skolem)
firstViolin(oslo, lie)
```



- Orchestras and Chamber ensembles are Ensembles.
- Chamber ensembles have only one instrument on each voice,
- in particular, only one first violin.

Restrictions, non-unique names and open worlds

Restrictions + the OWA and the NUNA can be tricky, consider:

TBox:

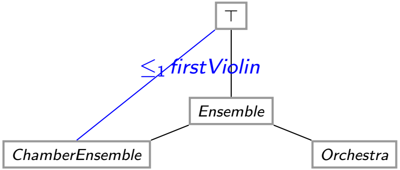
$$Orchestra \sqsubseteq Ensemble$$

$$ChamberEnsemble \sqsubseteq Ensemble$$

$$ChamberEnsemble \sqsubseteq \leq_1 firstViolin.\top$$

ABox:

```
Ensemble(oslo)
firstViolin(oslo, skolem)
firstViolin(oslo, lie)
```



- Orchestras and Chamber ensembles are Ensembles.
- Chamber ensembles have only one instrument on each voice,
- in particular, only one first violin.
- `oslo` has two first violins; is `oslo` an Orchestra?

Unexpected (non-)results

It does not follow from TBox + ABox that `oslo` is an *Orchestra*:

- An ensemble need neither be an orchestra nor a chamber ensemble, its "just" an ensemble.
- Add "covering axiom" $\mathit{Ensemble} \sqsubseteq \mathit{Orchestra} \sqcup \mathit{ChamberEnsemble}$:
  - An ensemble is an orchestra or a chamber ensemble.

Unexpected (non-)results

It does not follow from TBox + ABox that `oslo` is an *Orchestra*:

- An ensemble need neither be an orchestra nor a chamber ensemble, its "just" an ensemble.
- Add "covering axiom" $\mathit{Ensemble} \sqsubseteq \mathit{Orchestra} \sqcup \mathit{ChamberEnsemble}$:
  - An ensemble is an orchestra or a chamber ensemble.

It still does not follow that `oslo` is an *Orchestra*:

- This is due to the NUNA.
- We cannot assume that `skolem` and `lie` are distinct.
- The statement `skolem owl:differentFrom lie`, i.e., `skolem` $\neq$ `lie`, makes `oslo` an orchestra.

Unexpected (non-)results

It does not follow from TBox + ABox that `oslo` is an *Orchestra*:

- An ensemble need neither be an orchestra nor a chamber ensemble, its "just" an ensemble.
- Add "covering axiom" $\mathit{Ensemble} \sqsubseteq \mathit{Orchestra} \sqcup \mathit{ChamberEnsemble}$:
  - An ensemble is an orchestra or a chamber ensemble.

It still does not follow that `oslo` is an *Orchestra*:

- This is due to the NUNA.
- We cannot assume that `skolem` and `lie` are distinct.
- The statement `skolem owl:differentFrom lie`, i.e., `skolem` $\neq$ `lie`, makes `oslo` an orchestra.

If we remove `firstViolin(oslo, lie)`, is `oslo` a *ChamberEnsemble*?

Unexpected (non-)results

It does not follow from TBox + ABox that `oslo` is an *Orchestra*:

- An ensemble need neither be an orchestra nor a chamber ensemble, its “just” an ensemble.
- Add “covering axiom” $\mathit{Ensemble} \sqsubseteq \mathit{Orchestra} \sqcup \mathit{ChamberEnsemble}$:
  - An ensemble is an orchestra or a chamber ensemble.

It still does not follow that `oslo` is an *Orchestra*:

- This is due to the NUNA.
- We cannot assume that `skolem` and `lie` are distinct.
- The statement `skolem owl:differentFrom lie`, i.e., `skolem` $\neq$ `lie`, makes `oslo` an orchestra.

If we remove `firstViolin(oslo, lie)`, is `oslo` a *ChamberEnsemble*?

- it does not follow that `oslo` is a *ChamberEnsemble*.
- This is due to the OWA:
- `oslo` may have other first violinists.

Protégé demo of previous slide

- Make class Ensemble.
- Make subclass Orchestra.
- Make subclass ChamberEnsemble.
- Make object property firstViolin.
- Make `firstViolin max 1` superclass of ChamberEnsemble.
- Make an Ensemble oslo
- Make a Thing skolem
- Make a Thing lie
- Add firstViolin skolem to oslo
- Add firstViolin lie to oslo
- Classify! Nothing happens.
- Add covering axiom: `Orchestra or ChamberEnsemble` superclass of Ensemble.
- Classify! Nothing happens.
- `skolem` is different from `lie`
- Classify! Bingo! `oslo` is an Orchestra!

Role characteristics and relationships (RBox)

Vocabulary

Given the *roles* $\{R_1, R_2, \ldots\}$

Role characteristics and relationships (RBox)

Vocabulary

Given the *roles* $\{R_1, R_2, \ldots\}$

Role descriptions

$$
\begin{array}{rlcl}
R, S \rightarrow & R_i & \mid & \text{(atomic role)} \\
& \top_{role} & \mid & \text{(universal role)} \\
& \bot_{role} & \mid & \text{(bottom role)}
\end{array}
$$

Role characteristics and relationships (RBox)

Vocabulary

Given the *roles* $\{R_1, R_2, \ldots\}$

Role descriptions

$R, S \rightarrow$	R_i	$\mid$	(atomic role)
	$\top_{role}$	$\mid$	(universal role)
	$\bot_{role}$	$\mid$	(bottom role)
	$\neg R$	$\mid$	(complement role)
	R^{-}	$\mid$	(inverse role)
	$R \sqcap S$	$\mid$	(role intersection)
	$R \circ S$	$\mid$	(role chain)

Rbox (cont.)

- Role axioms: Let R and S be roles, then we can assert
  - subsumption: $R \sqsubseteq S$ $(R^{\mathcal{I}} \subseteq S^{\mathcal{I}})$,
  - equivalence: $R \equiv S$ $(R^{\mathcal{I}} = S^{\mathcal{I}})$,
  - disjointness: $R \sqcap S \sqsubseteq \bot_{\text{role}}$ $(R^{\mathcal{I}} \cap S^{\mathcal{I}} \subseteq \emptyset)$,
  - key: R is a *key* for concept C.

[2] Restrictions apply

Rbox (cont.)

- Role axioms: Let R and S be roles, then we can assert
  - subsumption: $R \sqsubseteq S$ $(R^{\mathcal{I}} \subseteq S^{\mathcal{I}})$,
  - equivalence: $R \equiv S$ $(R^{\mathcal{I}} = S^{\mathcal{I}})$,
  - disjointness: $R \sqcap S \sqsubseteq \bot_{\text{role}}$ $(R^{\mathcal{I}} \cap S^{\mathcal{I}} \subseteq \emptyset)$,
  - key: R is a *key* for concept C.
- A role can have the characteristics (axioms):
  - reflexive, irreflexive,
  - symmetric, asymmetric,
  - transitive, or/and[2]
  - functional, inverse functional.

[2] Restrictions apply

New roles

- The universal role, and the empty role—for both object roles and data roles.
- Syntax:
  - (DL: U (universal object role), D (universal data value role))
  - RDF/OWL, Manchester: `owl:topObjectProperty`, `owl:topDataProperty`, `owl:bottomObjectProperty`, `owl:bottomDataProperty`

New roles

- The universal role, and the empty role—for both object roles and data roles.
- Syntax:
  - (DL: U (universal object role), D (universal data value role))
  - RDF/OWL, Manchester: `owl:topObjectProperty`, `owl:topDataProperty`, `owl:bottomObjectProperty`, `owl:bottomDataProperty`
- Semantics:
  - $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\mathcal{D}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Lambda$

New roles

- The universal role, and the empty role—for both object roles and data roles.
- Syntax:
  - (DL: U (universal object role), D (universal data value role))
  - RDF/OWL, Manchester: `owl:topObjectProperty`, `owl:topDataProperty`, `owl:bottomObjectProperty`, `owl:bottomDataProperty`
- Semantics:
  - $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\mathcal{D}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Lambda$
- Reads:
  - all pairs of individuals are connected by `owl:topObjectProperty`,
  - no individuals are connected by `owl:bottomObjectProperty`.
  - all possible individuals are connected with all literals by `owl:topDataProperty`,
  - no individual is connected by `owl:bottomDataProperty` to a literal.

Corresponding mathematical properties and operations

If R and S are binary relations on X then

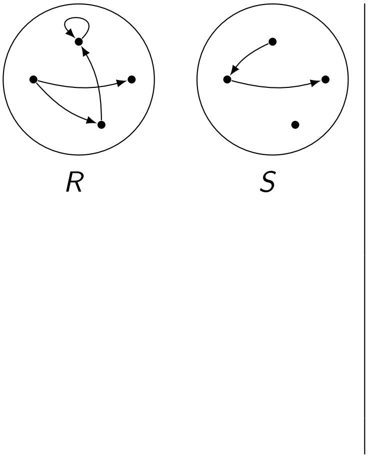
- $(R^{-})^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \mid \langle b^{\mathcal{I}}, a^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$

Corresponding mathematical properties and operations

If R and S are binary relations on X then

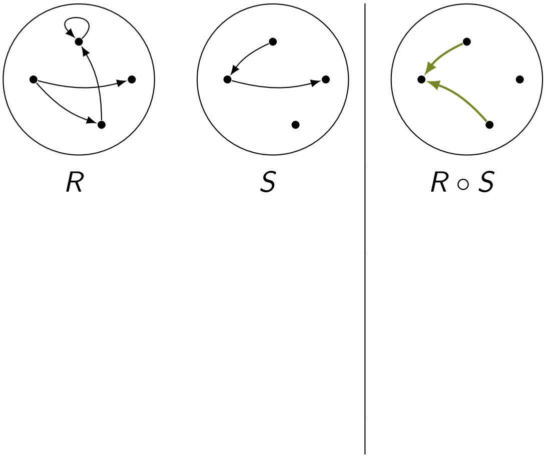
- $(R^{-})^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \mid \langle b^{\mathcal{I}}, a^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$
- $(R \circ S)^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, c^{\mathcal{I}}\rangle \mid \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}, \langle b^{\mathcal{I}}, c^{\mathcal{I}}\rangle \in S^{\mathcal{I}}\}$

Role chaining and inverses illustrated

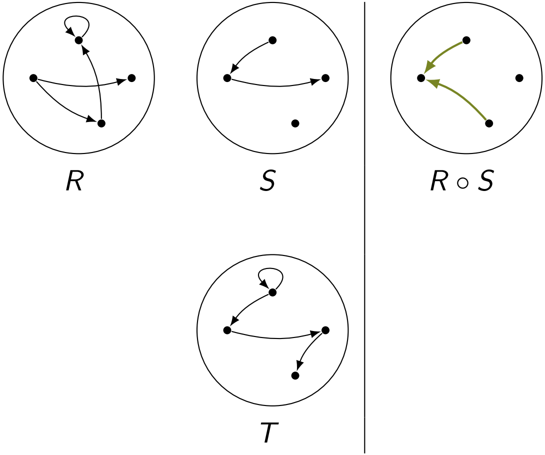


R S

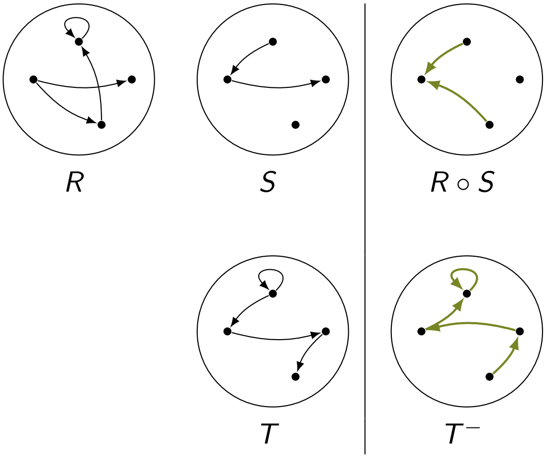
Role chaining and inverses illustrated



Role chaining and inverses illustrated



Role chaining and inverses illustrated



Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive: if $\langle a, a \rangle \in R$ for all $a \in X$ ($X \sqsubseteq \exists R.Self$)

Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in X$	$(X \sqsubseteq \exists R.Self)$
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	$(X \sqsubseteq \neg\exists R.Self)$

Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in X$	$(X \sqsubseteq \exists R.Self)$
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	$(X \sqsubseteq \neg\exists R.Self)$
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	$(R^{-} \sqsubseteq R)$

Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in X$	$(X \sqsubseteq \exists R.Self)$
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	$(X \sqsubseteq \neg\exists R.Self)$
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	$(R^- \sqsubseteq R)$
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	$(R^- \sqsubseteq \neg R)$

Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in X$	($X \sqsubseteq \exists R.Self$)
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	($X \sqsubseteq \neg\exists R.Self$)
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	($R^- \sqsubseteq R$)
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	($R^- \sqsubseteq \neg R$)
Transitive:	if $\langle a, b \rangle, \langle b, c \rangle \in R$ implies $\langle a, c \rangle \in R$	($R \circ R \sqsubseteq R$)

Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in X$	$(X \sqsubseteq \exists R.Self)$
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	$(X \sqsubseteq \neg\exists R.Self)$
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	$(R^- \sqsubseteq R)$
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	$(R^- \sqsubseteq \neg R)$
Transitive:	if $\langle a, b \rangle, \langle b, c \rangle \in R$ implies $\langle a, c \rangle \in R$	$(R \circ R \sqsubseteq R)$
Functional:	if $\langle a, b \rangle, \langle a, c \rangle \in R$ implies $b = c$	$(\top \sqsubseteq \leq_1 R.\top)$

Common properties of roles

A relation R over a set X ($R \subseteq X \times X$) is

Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in X$	$(X \sqsubseteq \exists R.Self)$
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	$(X \sqsubseteq \neg\exists R.Self)$
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	$(R^{-} \sqsubseteq R)$
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	$(R^{-} \sqsubseteq \neg R)$
Transitive:	if $\langle a, b \rangle, \langle b, c \rangle \in R$ implies $\langle a, c \rangle \in R$	$(R \circ R \sqsubseteq R)$
Functional:	if $\langle a, b \rangle, \langle a, c \rangle \in R$ implies $b = c$	$(\top \sqsubseteq \leq_1 R.\top)$
Inverse functional:	if $\langle a, b \rangle, \langle c, b \rangle \in R$ implies $a = c$	$(\top \sqsubseteq \leq_1 R^{-}.\top)$

Properties in OWL

Remember: three kinds of *mutually disjoint* properties in OWL:

1. `owl:DatatypeProperty`
   - link individuals to data values, e.g., `xsd:string`.
   - Examples: `:hasAge`, `:hasSurname`.
2. `owl:ObjectProperty`
   - link individuals to individuals.
   - Example: `:hasFather`, `:driveAxle`.
3. `owl:AnnotationProperty`
   - has no logical implication, ignored by reasoners.
   - Examples: `rdfs:label`, `dc:creator`.

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
    - reflexive—or they would not be datatype properties,
    - transitive—since no property takes data values in 1. position,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,
  - inverses—as above,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,
  - inverses—as above,
  - inverse functional—for computational reasons,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
    - reflexive—or they would not be datatype properties,
    - transitive—since no property takes data values in 1. position,
    - symmetric—as above,
    - inverses—as above,
    - inverse functional—for computational reasons,
    - part of chains—as above,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,
  - inverses—as above,
  - inverse functional—for computational reasons,
  - part of chains—as above,
  - so, what remains is: functionality,

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
    - reflexive—or they would not be datatype properties,
    - transitive—since no property takes data values in 1. position,
    - symmetric—as above,
    - inverses—as above,
    - inverse functional—for computational reasons,
    - part of chains—as above,
    - so, what remains is: functionality,
    - (and subsumption, equivalence and disjointness).

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,
  - inverses—as above,
  - inverse functional—for computational reasons,
  - part of chains—as above,
  - so, what remains is: functionality,
  - (and subsumption, equivalence and disjointness).

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,
  - inverses—as above,
  - inverse functional—for computational reasons,
  - part of chains—as above,
  - so, what remains is: functionality,
  - (and subsumption, equivalence and disjointness).
- (Annotation properties have no logical implication, so nothing can be said about them.)

Some relations from ordinary language

- Symmetric relations:

Some relations from ordinary language

- Symmetric relations:
    - *hasSibling*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*
  - *hasSibling*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*
  - *hasSibling*
- Functional relations:

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*
  - *hasSibling*
- Functional relations:
  - *hasBiologicalMother*

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*
  - *hasSibling*
- Functional relations:
  - *hasBiologicalMother*
- Inverse functional relations:

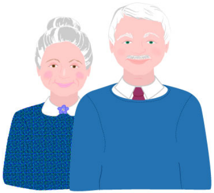
Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*
  - *hasSibling*
- Functional relations:
  - *hasBiologicalMother*
- Inverse functional relations:
  - *gaveBirthTo*

Examples inverses and chains

Some inverses:

- $hasParent \equiv hasChild^{-}$
- $hasBiologicalMother \equiv gaveBirthTo^{-}$
- $olderThan \equiv youngerThan^{-}$



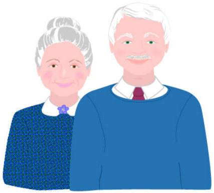
Examples inverses and chains

Some inverses:

- $hasParent \equiv hasChild^{-}$
- $hasBiologicalMother \equiv gaveBirthTo^{-}$
- $olderThan \equiv youngerThan^{-}$

Some role chains:

- $hasParent \circ hasParent \sqsubseteq hasGrandParent$
- $hasAncestor \circ hasAncestor \sqsubseteq hasAncestor$
- $hasParent \circ hasBrother \sqsubseteq hasUncle$



Quirks

Role modelling in OWL 2 can get excessively complicated.

- For instance:
  - transitive roles cannot be irreflexive or asymmetric,

Quirks

Role modelling in OWL 2 can get excessively complicated.

- For instance:
  - transitive roles cannot be irreflexive or asymmetric,
  - role inclusions are not allowed to cycle, i.e. not

$$\texttt{hasParent} \circ \texttt{hasHusband} \sqsubseteq \texttt{hasFather}$$
$$\texttt{hasFather} \sqsubseteq \texttt{hasParent}.$$

Quirks

Role modelling in OWL 2 can get excessively complicated.

- For instance:
  - transitive roles cannot be irreflexive or asymmetric,
  - role inclusions are not allowed to cycle, i.e. not

$$\texttt{hasParent} \circ \texttt{hasHusband} \sqsubseteq \texttt{hasFather}$$
$$\texttt{hasFather} \sqsubseteq \texttt{hasParent}.$$

  - transitive roles R and S cannot be declared disjoint

Quirks

Role modelling in OWL 2 can get excessively complicated.

- For instance:
  - transitive roles cannot be irreflexive or asymmetric,
  - role inclusions are not allowed to cycle, i.e. not

$$\texttt{hasParent} \circ \texttt{hasHusband} \sqsubseteq \texttt{hasFather}$$
$$\texttt{hasFather} \sqsubseteq \texttt{hasParent}.$$

  - transitive roles R and S cannot be declared disjoint
- Note:
  - these restrictions can be hard to keep track of
  - the reason they exist are computational, not logical

Quirks

Role modelling in OWL 2 can get excessively complicated.

- For instance:
  - transitive roles cannot be irreflexive or asymmetric,
  - role inclusions are not allowed to cycle, i.e. not

$$\texttt{hasParent} \circ \texttt{hasHusband} \sqsubseteq \texttt{hasFather}$$
$$\texttt{hasFather} \sqsubseteq \texttt{hasParent}.$$

  - transitive roles R and S cannot be declared disjoint
- Note:
  - these restrictions can be hard to keep track of
  - the reason they exist are computational, not logical
- Fortunately:
  - There are also *simple* patterns
  - that are quite useful.

Creating datatypes

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.

Creating datatypes

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.
- New datatypes can be defined by boolean operations: $\neg$, $\sqcap$, $\sqcup$:
  - `owl:datatypeComplementOf`, `owl:intersectionOf`, `owl:unionOf`.

Creating datatypes

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.
- New datatypes can be defined by boolean operations: $\neg$, $\sqcap$, $\sqcup$:
  - `owl:datatypeComplementOf`, `owl:intersectionOf`, `owl:unionOf`.
- Datatypes may be restricted with *constraining facets*, borrowed from XML Schema.
  - For numeric datatypes: `xsd:minInclusive`, `xsd:maxInclusive`
  - For string datatypes: `xsd:minLenght`, `xsd:maxLenght`, `xsd:pattern`.

Creating datatypes

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.
- New datatypes can be defined by boolean operations: $\neg$, $\sqcap$, $\sqcup$:
  - `owl:datatypeComplementOf`, `owl:intersectionOf`, `owl:unionOf`.
- Datatypes may be restricted with *constraining facets*, borrowed from XML Schema.
  - For numeric datatypes: `xsd:minInclusive`, `xsd:maxInclusive`
  - For string datatypes: `xsd:minLenght`, `xsd:maxLenght`, `xsd:pattern`.
- Example:
  - Teenager is equivalent to: (Manchester)
    `Person and (age some positiveInteger[>= 13, <= 19])`
  - "A teenager is a person of age 13 to 19."

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.

  ($\mathit{NuclearFam} \sqsubseteq =_2 \mathit{hasMember}.\mathit{Parent} \sqcap \geq_2 \mathit{hasMember}.\mathit{Child} \sqcap \exists \mathit{hasMember}.\mathit{Dog}$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.

  ($NuclearFam \sqsubseteq {=_2}\ hasMember.Parent \sqcap\ {\geq_2}\ hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa).

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends. ($hasFriend \circ hasFriend \sqsubseteq hasFriend$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends. ($hasFriend \circ hasFriend \sqsubseteq hasFriend$)
- ✓ If Homer is married to Marge, then Marge is married to Homer.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends. ($hasFriend \circ hasFriend \sqsubseteq hasFriend$)
- ✓ If Homer is married to Marge, then Marge is married to Homer. ($marriedTo^{-} \sqsubseteq marriedTo$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends. ($hasFriend \circ hasFriend \sqsubseteq hasFriend$)
- ✓ If Homer is married to Marge, then Marge is married to Homer. ($marriedTo^{-} \sqsubseteq marriedTo$)
- ✓ If Homer is a parent of Bart, then Bart is a child of Homer.

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends. ($hasFriend \circ hasFriend \sqsubseteq hasFriend$)
- ✓ If Homer is married to Marge, then Marge is married to Homer. ($marriedTo^- \sqsubseteq marriedTo$)
- ✓ If Homer is a parent of Bart, then Bart is a child of Homer. ($parentOf^- \sqsubseteq childOf$)

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
  ($NuclearFam \sqsubseteq =_2 hasMember.Parent \sqcap \geq_2 hasMember.Child \sqcap \exists hasMember.Dog$)
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ($\top \sqsubseteq \exists loves.\{mary\}$ or $Person \sqsubseteq \exists loves.\{mary\}$)
- ✓ Adam is not Eve (and vice versa). ($adam \neq eve$)
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle. ($hasFather \circ hasBrother \sqsubseteq hasUncle$)
- ✓ My friend's friends are also my friends. ($hasFriend \circ hasFriend \sqsubseteq hasFriend$)
- ✓ If Homer is married to Marge, then Marge is married to Homer. ($marriedTo^- \sqsubseteq marriedTo$)
- ✓ If Homer is a parent of Bart, then Bart is a child of Homer. ($parentOf^- \sqsubseteq childOf$)

... and more!

DL: Family of languages

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

Next week

- More modelling with OWL/OWL 2.
- What cannot be expressed in OWL/OWL 2?