

Reasoning and Planning

Unit 6. Terminological Reasoning

Pedro Cabalar

Dept. Computer Science
University of Corunna, SPAIN

November 8, 2025

Description Logic

- Domain knowledge uses many concepts. E.g. a piston is a mechanical piece and is part of the engine that is a device ...

Description Logic

- Domain knowledge uses many **concepts**. E.g. a **piston** **is a** **mechanical piece** and **is part of** the **engine** that **is a** **device** ...
- 👉 Represent **concept definitions** for a given domain = **Terminological Knowledge**
- In the past known as **terminological systems** or **concept languages**

Description Logic

- Domain knowledge uses many **concepts**. E.g. a **piston** **is a** **mechanical piece** and **is part of** the **engine** that **is a** **device** ...
- 👍 Represent **concept definitions** for a given domain = **Terminological Knowledge**
- In the past known as **terminological systems** or **concept languages**
- Nowadays, main use: description of **Ontologies** for the **Semantic Web** (OWL-DL, OWL-Lite).

Description Logic

- Domain knowledge uses many **concepts**. E.g. a **piston** **is a** **mechanical piece** and **is part of** the **engine** that **is a device** ...
- 👍 Represent **concept definitions** for a given domain = **Terminological Knowledge**
- In the past known as **terminological systems** or **concept languages**
- Nowadays, main use: description of **Ontologies** for the **Semantic Web** (OWL-DL, OWL-Lite).
- **Description Logic** (DL) uses basic operators (similar to modal logic) that can be translated to **predicate calculus**.

- **Syntax:** we start from
 - ① A set of atoms called **concepts**. They represent **classes** or **sets** of objects. E.g.: *Person*, *Mammal*, *Vehicle*, *Blue*, *English*...

- **Syntax:** we start from
 - ① A set of atoms called **concepts**. They represent **classes** or **sets** of objects. E.g.: *Person, Mammal, Vehicle, Blue, English...*
 - ② A set of **roles** that represent **binary relations** among objects. E.g.: *likes, owns, pays, travels_by, has_child ...*

- **Syntax:** we start from

- ① A set of atoms called **concepts**. They represent **classes** or **sets** of objects. E.g.: *Person, Mammal, Vehicle, Blue, English...*
- ② A set of **roles** that represent **binary relations** among objects. E.g.: *likes, owns, pays, travels_by, has_child ...*
- ③ A set of **constructors** to define new concepts recursively. E.g.: blue or red vehicles = $Vehicle \sqcap (Blue \sqcup Red)$.
E.g.: a father is a $Man \sqcap \exists has_child$

- **Syntax:** we start from
 - ① A set of atoms called **concepts**. They represent **classes** or **sets** of objects. E.g.: *Person, Mammal, Vehicle, Blue, English...*
 - ② A set of **roles** that represent **binary relations** among objects. E.g.: *likes, owns, pays, travels_by, has_child ...*
 - ③ A set of **constructors** to define new concepts recursively. E.g.: blue or red vehicles = $Vehicle \sqcap (Blue \sqcup Red)$.
E.g.: a father is a $Man \sqcap \exists has_child$
- There exists a whole **family** of Description Logics depending on the constructors we allow

Description Logic: \mathcal{FL}^-

- The simplest DL is \mathcal{FL}^- . Syntax:

$$C ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R \mid \top \mid \perp$$

where A =atomic concept, C, D =concepts and R =role.

- The simplest DL is \mathcal{FL}^- . Syntax:

$$C ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R \mid \top \mid \perp$$

where A =atomic concept, C, D =concepts and R =role.

- Alternative syntax:

$$C ::= A \mid (: \text{and } C D) \mid (: \text{all } R C) \mid (: \text{some } R)$$

- The simplest DL is \mathcal{FL}^- . Syntax:

$$C ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R \mid \top \mid \perp$$

where A =atomic concept, C, D =concepts and R =role.

- Alternative syntax:

$$C ::= A \mid (: \text{and } C D) \mid (: \text{all } R C) \mid (: \text{some } R)$$

- Quantifiers: $\forall \text{has_child.Female}$ are those living beings whose offsprings are all female, whereas $\exists \text{has_child}$ are those that have, at list, some child.

- The simplest DL is \mathcal{FL}^- . Syntax:

$$C ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R \mid \top \mid \perp$$

where A =atomic concept, C, D =concepts and R =role.

- Alternative syntax:

$$C ::= A \mid (: \text{and } C D) \mid (: \text{all } R C) \mid (: \text{some } R)$$

- Quantifiers: $\forall \text{has_child.Female}$ are those living beings whose offsprings are all female, whereas $\exists \text{has_child}$ are those that have, at list, some child. We can write it as $\exists \text{has_child}.\top$ too

Definition (Interpretation)

An interpretation \mathcal{I} is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where

- $\Delta^{\mathcal{I}}$ is a non-empty set called the **domain**
- $\cdot^{\mathcal{I}}$ is a function that maps:
 - 1 Each concept to a subset of $\Delta^{\mathcal{I}}$.
 - 2 Each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

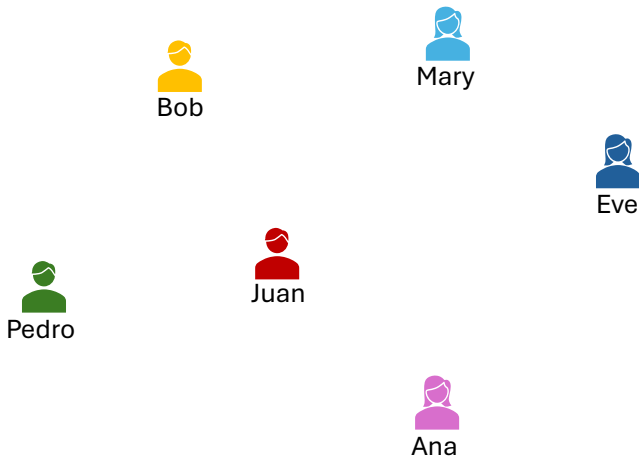
Definition (Interpretation)

An interpretation \mathcal{I} is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where

- $\Delta^{\mathcal{I}}$ is a non-empty set called the **domain**
- $\cdot^{\mathcal{I}}$ is a function that maps:
 - 1 Each concept to a subset of $\Delta^{\mathcal{I}}$.
 - 2 Each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

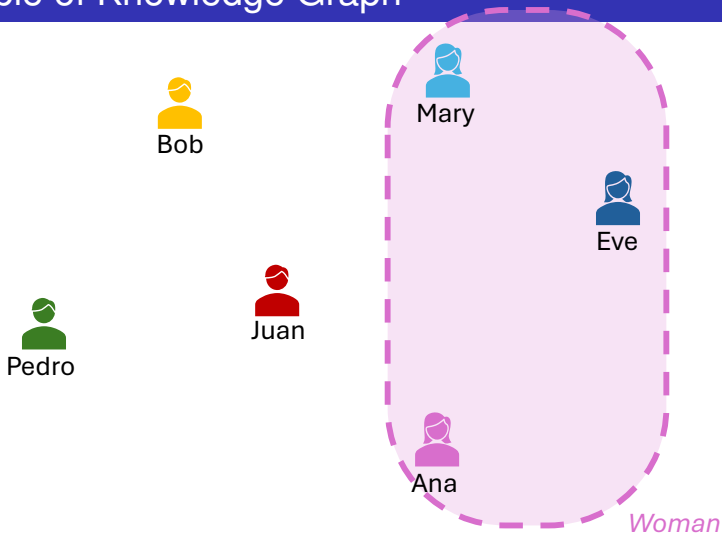
An interpretation can be represented as a **Knowledge Graph** ...

Example of Knowledge Graph



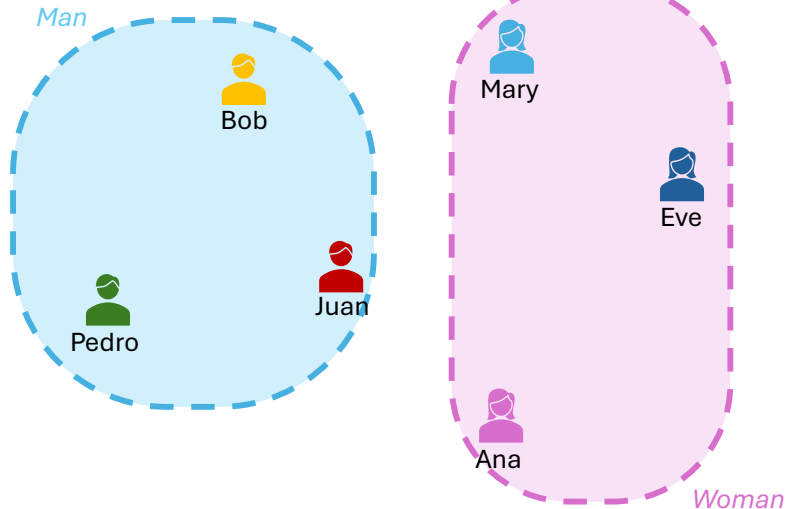
The domain $\Delta^{\mathcal{I}}$ consists of these 6 persons

Example of Knowledge Graph



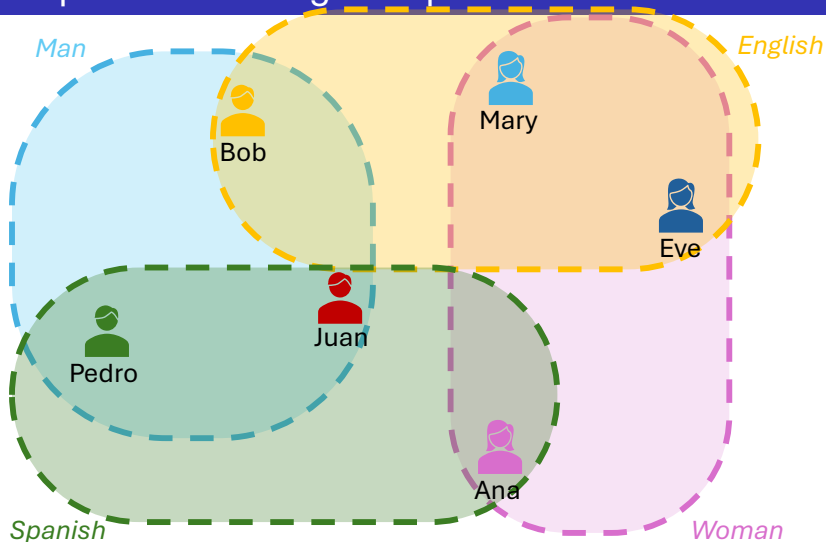
Concept *Woman^I* is the set of 3 women on the right

Example of Knowledge Graph



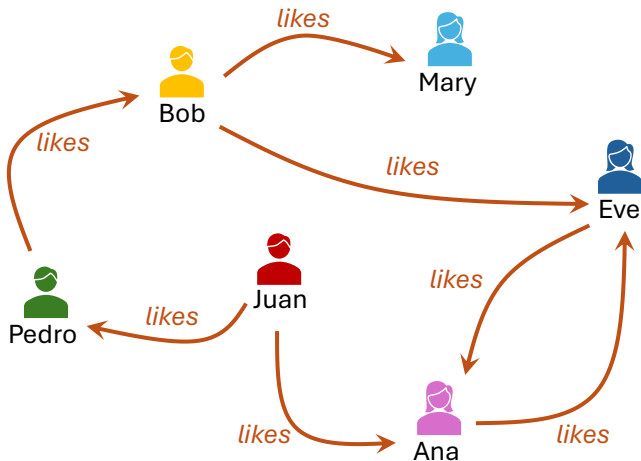
Concept Man^I is the set of 3 men on the left

Example of Knowledge Graph



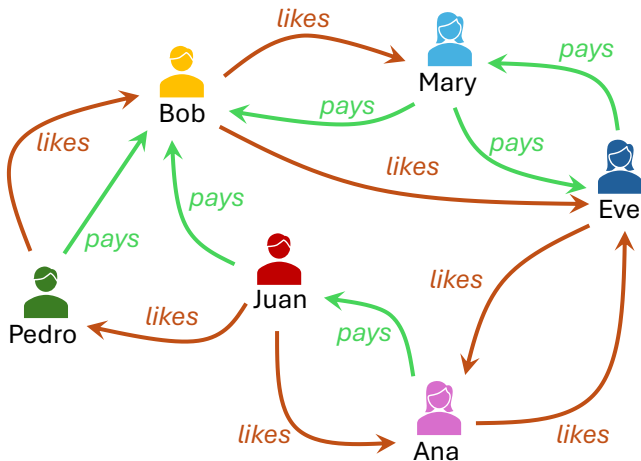
Concepts *Spanish^T* and *English^T* are these sets

Example of Knowledge Graph



Relations are pairs: for instance *likes* ^{\mathcal{I}} ...

Example of Knowledge Graph



Relations are pairs: another example $pays^T \dots$

- We extend $\cdot^{\mathcal{I}}$ for evaluation of non-atomic concepts:

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

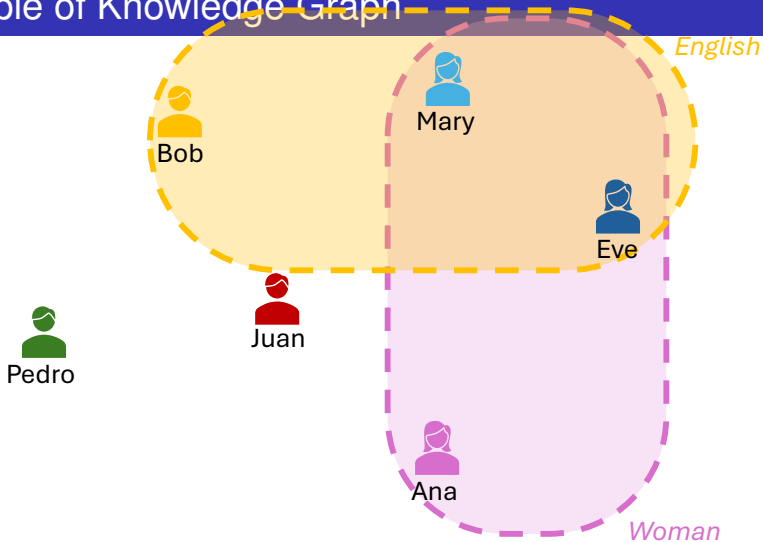
$$(\exists R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} : \exists y.(x, y) \in R^{\mathcal{I}}\}$$

$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} : \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$$

$$(\top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

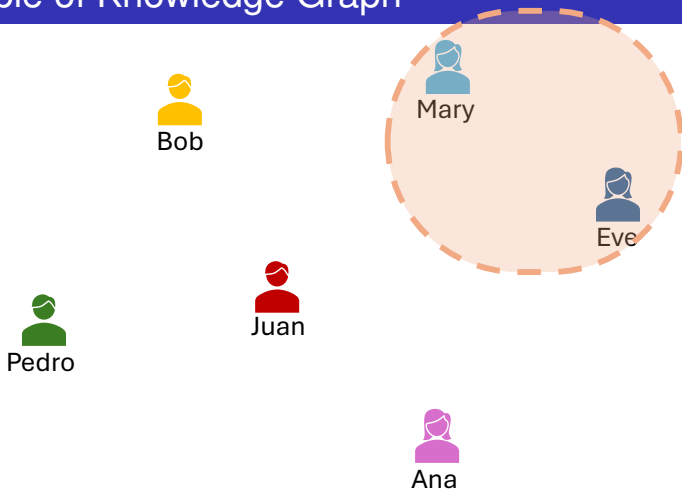
$$(\perp)^{\mathcal{I}} = \emptyset$$

Example of Knowledge Graph



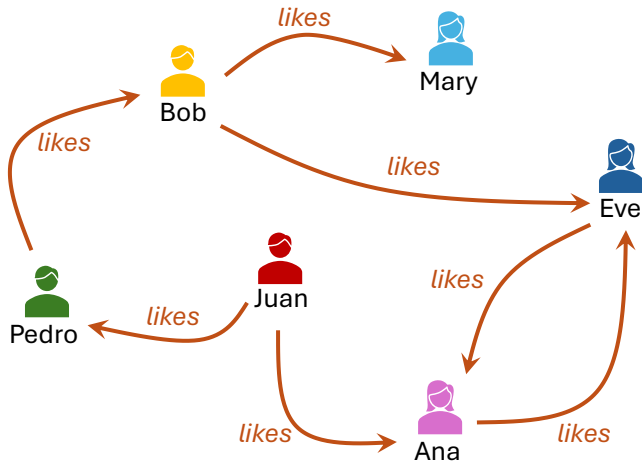
The concept *English* \cap *Woman* is the intersection set $\{Mary, Eve\}$

Example of Knowledge Graph



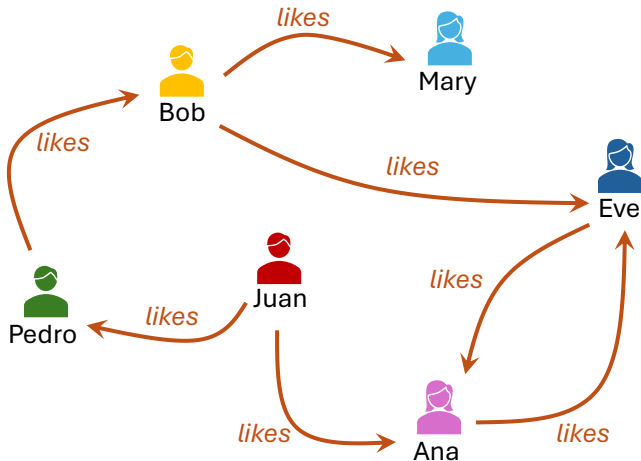
The concept *English* \cap *Woman* is the intersection set $\{Mary, Eve\}$

Example of Knowledge Graph



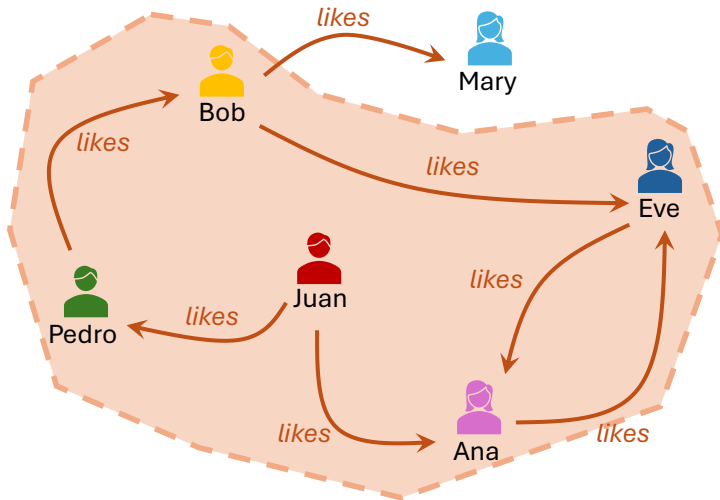
Concept \exists likes =???

Example of Knowledge Graph



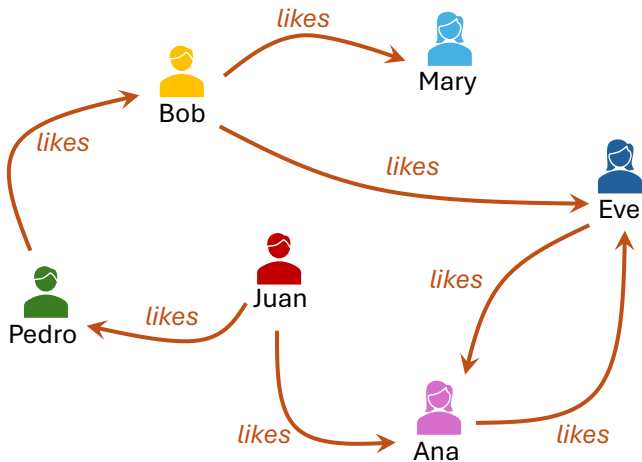
Concept \exists likes = people who like someone

Example of Knowledge Graph



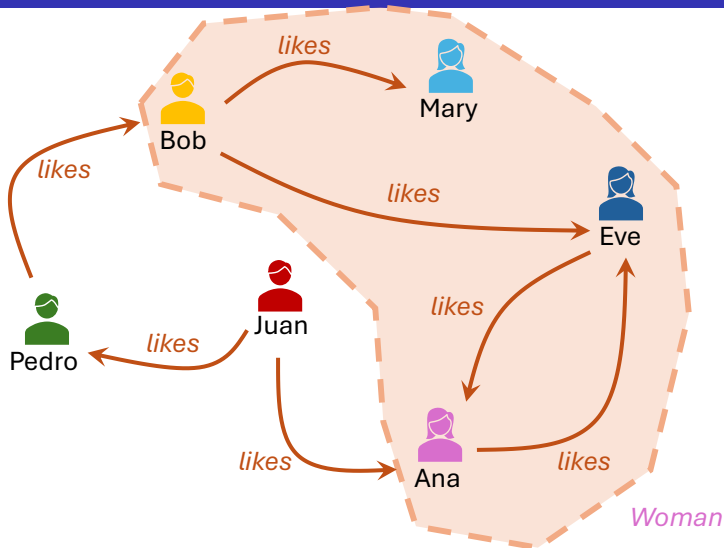
Concept \exists likes = people who like someone

Example of Knowledge Graph



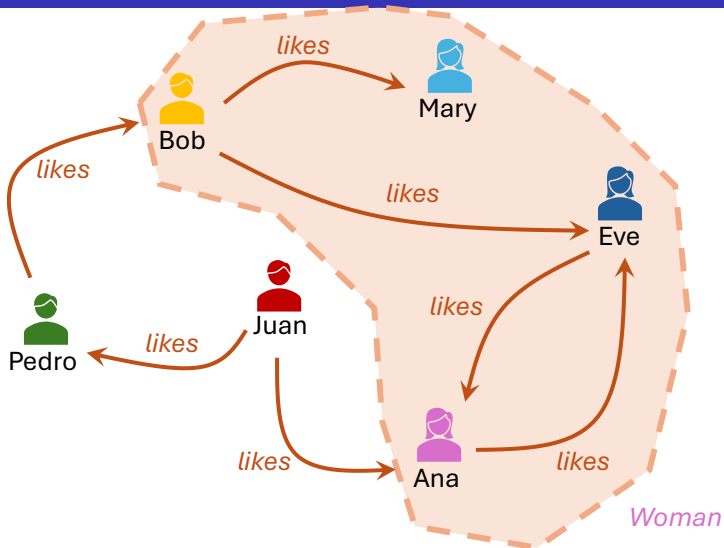
Concept $\forall \text{likes. Woman} = ???$

Example of Knowledge Graph



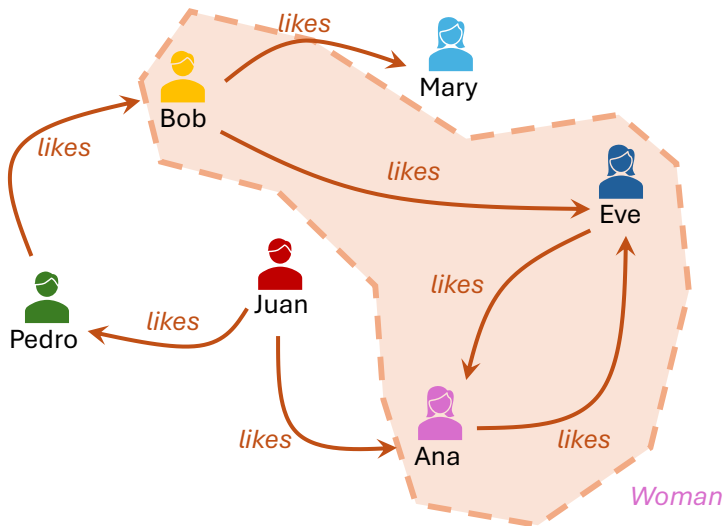
Concept $\forall \text{likes}. \text{Woman}$ = everybody they like are women

Example of Knowledge Graph



$\forall \text{likes.Woman}$ Note that **Mary** is included too!

Example of Knowledge Graph



$\forall \text{likes. Woman} \sqcap \exists \text{like}$ forces existence

Examples: which is the meaning of the following expressions?

Adult \sqcap *Male*

Examples: which is the meaning of the following expressions?

Adult \sqcap *Male*

$\forall has_child.(Adult \sqcap Male)$

Examples: which is the meaning of the following expressions?

$Adult \sqcap Male$

$\forall has_child.(Adult \sqcap Male)$

$\exists has_child \sqcap \forall has_child.(\exists has_child \sqcap Adult)$

- A second example: \mathcal{ALC} is more expressive than \mathcal{FL}^- . It allows:

$$C ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R \mid \neg C$$

$$\text{con } (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

- A second example: \mathcal{ALC} is more expressive than \mathcal{FL}^- . It allows:

$$C ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R \mid \neg C$$

$$\text{con } (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

- A more limited variant \mathcal{AL} replaces $\neg C$ by $\neg A$ (only **atomic concepts** can be negated).

Description Logics: most common constructors

Constructor	Sintaxis	Semántica
Concept assertion	C	$C^{\mathcal{I}} \in \mathcal{C}^{\mathcal{I}}$
Role assertion	$(o_1, o_2) : R$	$(o_1^{\mathcal{I}}, o_2^{\mathcal{I}}) \in R^{\mathcal{I}}$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Existential	$\exists R.C$	$\{x : \exists y.(x, y) \in R^{\mathcal{I}} \& y \in C^{\mathcal{I}}\}$
Universal	$\forall R.C$	$\{x : \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
Cardinality	$(\geq n R)$	$\{x : \{y.(x, y) \in R^{\mathcal{I}}\} \geq n\}$
	$(\leq n R)$	$\{x : \{y.(x, y) \in R^{\mathcal{I}}\} \leq n\}$
Inverse	R^{-}	$\{(x, y) : (y, x) \in R^{\mathcal{I}}\}$
Transitive	R^*	$(R^{\mathcal{I}})^*$
Enumeration (one-of)	$\{o_1, \dots, o_n\}$	$\{o_1^{\mathcal{I}}, \dots, o_n^{\mathcal{I}}\}$

...

Description Logics: naming

- To name each combination, we use the **initials**

\mathcal{F} functional property ($\leq 1 R$)

\mathcal{E} general existential $\exists R.C$

\mathcal{U} concept union $C \sqcup D$

\mathcal{C} concept negation $\neg C$

$\mathcal{S} = \mathcal{ALC}$ plus transitive roles R^*

\mathcal{H} role hierarchy

\mathcal{R} reflexive, irreflexive and disjoint roles

\mathcal{I} inverse role R^-

\mathcal{O} enumeration (one-of) $\{o_1, \dots, o_n\}$

\mathcal{N} cardinality constraints ($\geq n R$), ($\leq n R$)

\mathcal{Q} qualified cardinality constraints ($\geq n R.C$), ($\leq n R.C$)

(\mathcal{D}) data types (integer, string, etc)

Description Logics: naming

- To name each combination, we use the **initials**
 - \mathcal{F} functional property ($\leq 1 R$)
 - \mathcal{E} general existential $\exists R.C$
 - \mathcal{U} concept union $C \sqcup D$
 - \mathcal{C} concept negation $\neg C$
 - $\mathcal{S} = \mathcal{ALC}$ plus transitive roles R^*
 - \mathcal{H} role hierarchy
 - \mathcal{R} reflexive, irreflexive and disjunct roles
 - \mathcal{I} inverse role R^-
 - \mathcal{O} enumeration (one-of) $\{o_1, \dots, o_n\}$
 - \mathcal{N} cardinality constraints ($\geq n R$), ($\leq n R$)
 - \mathcal{Q} qualified cardinality constraints ($\geq n R.C$), ($\leq n R.C$)
 - (\mathcal{D}) data types (integer, string, etc)
- **OWL-DL** corresponds to $\mathcal{SHOIN}^{(\mathcal{D})}$ whereas **OWL-Lite** is based on $\mathcal{SHIF}^{(\mathcal{D})}$.

Description Logic

- A (terminological) **Knowledge Base** consists of two sets of expressions **TBox** and **ABox**.

Description Logic

- A (terminological) **Knowledge Base** consists of two sets of expressions **TBox** and **ABox**.
- **TBox**: contains general terminological declarations. Two types:
 - 1 A **concept definition** $A \equiv C$. Examples:

Woman \equiv *Person* \sqcap *Female*

Mother \equiv *Woman* \sqcap \exists *has_child*

(**acyclicity** is usually required)

- A (terminological) **Knowledge Base** consists of two sets of expressions **TBox** and **ABox**.
- **TBox**: contains general terminological declarations. Two types:
 - 1 A **concept definition** $A \equiv C$. Examples:

$Woman \equiv Person \sqcap Female$

$Mother \equiv Woman \sqcap \exists has_child$

(**acyclicity** is usually required)

- 2 An **inclusion axiom** $C_1 \sqsubseteq C_2$. Example

$\exists has_child.Person \sqsubseteq Person$

They impose constraints on our model

- **ABox**: contains assertions about specific element and relation instances in the domain. Two types:
 - 1 Concept assertion $o : C$. Example:

Moby_Dick : Whale

Mary : Female \sqcap \exists has_child

- **ABox**: contains assertions about specific element and relation instances in the domain. Two types:
 - 1 Concept assertion $o : C$. Example:

Moby_Dick : Whale

Mary : Female \sqcap \exists has_child

- 2 Role assertions $(o_1, o_2) : R$. Example:

(Mary, Jesus) : has_child

Description Logic: reasoning

- Typical reasoning problems to solve. Given a Knowledge Base:
 - 1 Subsumption. Check whether a concept is more general than another $C \sqsubseteq D$

Description Logic: reasoning

- Typical **reasoning problems** to solve. Given a Knowledge Base:
 - 1 **Subsumption**. Check whether a concept is more general than another $C \sqsubseteq D$
 - 2 **Equivalence**. Check whether two concepts are equivalent $C \equiv D$

Description Logic: reasoning

- Typical reasoning problems to solve. Given a Knowledge Base:
 - 1 Subsumption. Check whether a concept is more general than another $C \sqsubseteq D$
 - 2 Equivalence. Check whether two concepts are equivalent $C \equiv D$
 - 3 Consistency. Check whether a concept has at least some model $C \equiv \perp$

Description Logic: reasoning

- Typical reasoning problems to solve. Given a Knowledge Base:
 - 1 Subsumption. Check whether a concept is more general than another $C \sqsubseteq D$
 - 2 Equivalence. Check whether two concepts are equivalent $C \equiv D$
 - 3 Consistency. Check whether a concept has at least some model $C \equiv \perp$
 - 4 Belonging. Check whether an individual is member of a concept $o : C$

Description Logic: reasoning

- Typical **reasoning problems** to solve. Given a Knowledge Base:
 - 1 **Subsumption**. Check whether a concept is more general than another $C \sqsubseteq D$
 - 2 **Equivalence**. Check whether two concepts are equivalent $C \equiv D$
 - 3 **Consistency**. Check whether a concept has at least some model $C \equiv \perp$
 - 4 **Belonging**. Check whether an individual is member of a concept $o : C$
- All these problems can be **reduced** to consistency. Example:
 $C \sqsubseteq D$ can be checked as the consistency test $C \sqcap \neg D \equiv \perp$.

Description Logic: reasoning

- There exist efficient methods to solve these reasoning problems
- Normally, **more expressive** language variant \Rightarrow **more complex** associated reasoning

Description Logic: reasoning

- There exist efficient methods to solve these reasoning problems
- Normally, **more expressive** language variant \Rightarrow **more complex** associated reasoning
- Example: subsumption in \mathcal{FL}^- is decidable in time complexity **P**.
- See Description Logic Complexity Navigator
<http://www.cs.man.ac.uk/~ezolin/dl/>

Description Logic: reasoning

- There exist efficient methods to solve these reasoning problems
- Normally, **more expressive** language variant \Rightarrow **more complex** associated reasoning
- Example: subsumption in \mathcal{FL}^- is decidable in time complexity P .
- See Description Logic Complexity Navigator
<http://www.cs.man.ac.uk/~ezolin/dl/>
- See protégé webpage
<https://protege.stanford.edu/>

Description Logic: translation to First-Order Logic

- Most Description Logics are reducible to **decidable** fragments of First-Order Logic

Description Logic: translation to First-Order Logic

- Most Description Logics are reducible to **decidable** fragments of First-Order Logic
- Each concept C becomes a unary predicate $C(x)$; each role R a binary predicate $R(x, y)$.

Description Logic: translation to First-Order Logic

- Most Description Logics are reducible to **decidable** fragments of First-Order Logic
- Each concept C becomes a unary predicate $C(x)$; each role R a binary predicate $R(x, y)$.
- We can use First-Order Logic with **variables**:

$$\begin{array}{ll} t_x(A) = A(x) & t_y(A) = A(y) \\ t_x(C \sqcap D) = t_x(C) \wedge t_x(D) & t_y(C \sqcap D) = t_y(C) \wedge t_y(D) \\ t_x(C \sqcup D) = t_x(C) \vee t_x(D) & t_y(C \sqcup D) = t_y(C) \vee t_y(D) \\ t_x(\exists R.C) = \exists y. R(x, y) \wedge t_y(C) & t_y(\exists R.C) = \exists x. R(y, x) \wedge t_x(C) \\ t_x(\forall R.C) = \forall y. R(x, y) \rightarrow t_y(C) & t_y(\forall R.C) = \forall x. R(y, x) \rightarrow t_x(C) \end{array}$$

Description Logic: translation to First-Order Logic

- Most Description Logics are reducible to **decidable** fragments of First-Order Logic
- Each concept C becomes a unary predicate $C(x)$; each role R a binary predicate $R(x, y)$.
- We can use First-Order Logic with **variables**:

$$\begin{array}{ll} t_x(A) = A(x) & t_y(A) = A(y) \\ t_x(C \sqcap D) = t_x(C) \wedge t_x(D) & t_y(C \sqcap D) = t_y(C) \wedge t_y(D) \\ t_x(C \sqcup D) = t_x(C) \vee t_x(D) & t_y(C \sqcup D) = t_y(C) \vee t_y(D) \\ t_x(\exists R.C) = \exists y. R(x, y) \wedge t_y(C) & t_y(\exists R.C) = \exists x. R(y, x) \wedge t_x(C) \\ t_x(\forall R.C) = \forall y. R(x, y) \rightarrow t_y(C) & t_y(\forall R.C) = \forall x. R(y, x) \rightarrow t_x(C) \end{array}$$

- In a TBox, we translate $C \equiv D$ into $\forall x. t_x(C) \leftrightarrow t_x(D)$.

Description Logic: translation to First-Order Logic

- An example

$t_x(\forall has_child.(Adult \sqcap Male))$

Description Logic: translation to First-Order Logic

- An example

$$\begin{aligned} & t_x(\forall has_child.(Adult \sqcap Male)) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male) \end{aligned}$$

- An example

$$\begin{aligned} & t_x(\forall has_child.(Adult \sqcap Male)) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult) \wedge t_y(Male) \end{aligned}$$

Description Logic: translation to First-Order Logic

- An example

$$t_x(\forall has_child.(Adult \sqcap Male))$$
$$= \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male)$$
$$= \forall y.has_child(x, y) \rightarrow t_y(Adult) \wedge t_y(Male)$$
$$= \forall y.has_child(x, y) \rightarrow Adult(y) \wedge Male(y)$$

Description Logic: translation to First-Order Logic

- An example

$$\begin{aligned} & t_x(\forall has_child.(Adult \sqcap Male)) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult) \wedge t_y(Male) \\ = & \forall y.has_child(x, y) \rightarrow Adult(y) \wedge Male(y) \end{aligned}$$

- Second example:

$$t_x(\exists has_child \sqcap \forall has_child.(\exists has_child \sqcap Adult))$$

Description Logic: translation to First-Order Logic

- An example

$$\begin{aligned} & t_x(\forall has_child.(Adult \sqcap Male)) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult) \wedge t_y(Male) \\ = & \forall y.has_child(x, y) \rightarrow Adult(y) \wedge Male(y) \end{aligned}$$

- Second example:

$$\begin{aligned} & t_x(\exists has_child \sqcap \forall has_child.(\exists has_child \sqcap Adult)) \\ = & (\exists y.has_child(x, y)) \wedge \\ & (\forall y.has_child(x, y) \rightarrow (\exists x.has_child(y, x)) \wedge Adult(y)) \end{aligned}$$

Description Logic: translation to First-Order Logic

- An example

$$\begin{aligned} & t_x(\forall has_child.(Adult \sqcap Male)) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult) \wedge t_y(Male) \\ = & \forall y.has_child(x, y) \rightarrow Adult(y) \wedge Male(y) \end{aligned}$$

- Second example:

$$\begin{aligned} & t_x(\exists has_child \sqcap \forall has_child.(\exists has_child \sqcap Adult)) \\ = & (\exists y.has_child(x, y)) \wedge \\ & (\forall y.has_child(x, y) \rightarrow (\exists x.has_child(y, x)) \wedge Adult(y)) \end{aligned}$$

Notice the difference

$$t_x(\exists has_child \sqcap \forall has_child.(\exists has_child.Adult))$$

Description Logic: translation to First-Order Logic

- An example

$$\begin{aligned} & t_x(\forall has_child.(Adult \sqcap Male)) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult \sqcap Male) \\ = & \forall y.has_child(x, y) \rightarrow t_y(Adult) \wedge t_y(Male) \\ = & \forall y.has_child(x, y) \rightarrow Adult(y) \wedge Male(y) \end{aligned}$$

- Second example:

$$\begin{aligned} & t_x(\exists has_child \sqcap \forall has_child.(\exists has_child \sqcap Adult)) \\ = & (\exists y.has_child(x, y)) \wedge \\ & (\forall y.has_child(x, y) \rightarrow (\exists x.has_child(y, x)) \wedge Adult(y)) \end{aligned}$$

Notice the difference

$$\begin{aligned} & t_x(\exists has_child \sqcap \forall has_child.(\exists has_child.Adult)) \\ = & (\exists y.has_child(x, y)) \wedge \\ & (\forall y.has_child(x, y) \rightarrow (\exists x.has_child(y, x) \wedge Adult(x))) \end{aligned}$$

Difference with respect to ASP/Logic Programming

- Unlike ASP, Prolog or SQL, Description Logics **do not make** the Closed World Assumption.

- Unlike ASP, Prolog or SQL, Description Logics **do not make** the **Closed World Assumption**.

Anything not asserted is just unknown (we have models in which it holds and models where it does not)

Difference with respect to ASP/Logic Programming

- Unlike ASP, Prolog or SQL, Description Logics **do not make** the **Closed World Assumption**.

Anything not asserted is just unknown (we have models in which it holds and models where it does not)

- We **do not have** the **Unique Names Assumption** either. Two different constants may denote the same individual.