ulm university universität uulm

Pavel Klinov[1] Bijan Parsia[2]

[1]University of Ulm [2]University of Manchester |

August 5, 2013

Practical Reasoning in DL (Introduction)

What's in this course?

1. Introduction (today)
   - Origins, syntax and semantics of basic DLs ($\mathcal{ALC}$)
   - Basic reasoning problems, inter-reducibility
   - Anatomy of a reasoner
2. Tableau algorithms
3. Classification and realization (Bijan)
4. Consequence-based reasoning for lightweight DL ($\mathcal{EL}$)
5. Dealing with data (ontology-based access to databases, Bijan)

What's in this course?

1. Introduction (today)
   - Origins, syntax and semantics of basic DLs ($\mathcal{ALC}$)
   - Basic reasoning problems, inter-reducibility
   - Anatomy of a reasoner
2. Tableau algorithms
3. Classification and realization (Bijan)
4. Consequence-based reasoning for lightweight DL ($\mathcal{EL}$)
5. Dealing with data (ontology-based access to databases, Bijan)

We'll pay special attention to practicality

- Essential optimization techniques
- Performance evaluation techniques

Welcome!

Let us know if you

- have questions; do ask them at any time
- have difficulties understanding
- find this course too slow/boring
- find this course too fast/difficult

In this course, we will:

- ask you to think a lot
- ask you to work through numerous examples

Origins of Description Logics

Basics: Syntax, Semantics, Reasoning Problems

Anatomy of a Reasoner

DLs: Where They Come From

DLs are logic-based knowledge representation (KR) formalisms

DLs: Where They Come From

DLs are logic-based knowledge representation (KR) formalisms

- Common perception: logic is difficult for human conception
  - e.g., how long does it take you to read $\forall x \exists y \forall z((r(x, y) \wedge s(y, z)) \Rightarrow (\neg s(a, y) \vee r(x, z)))$
  - or check that it is equivalent to $\forall x \exists y \forall z(r(x, z) \vee \neg r(x, y) \vee \neg s(y, z) \vee \neg s(a, y))$

DLs: Where They Come From

DLs are logic-based knowledge representation (KR) formalisms

- Common perception: logic is difficult for human conception
  - e.g., how long does it take you to read $\forall x \exists y \forall z ((r(x,y) \wedge s(y,z)) \Rightarrow (\neg s(a,y) \vee r(x,z)))$
  - or check that it is equivalent to $\forall x \exists y \forall z (r(x,z) \vee \neg r(x,y) \vee \neg s(y,z) \vee \neg s(a,y))$
- Also not-so-easy for machines (only semi-decidable)

DLs: Where They Come From

DLs are logic-based knowledge representation (KR) formalisms

- Common perception: logic is difficult for human conception
  - e.g., how long does it take you to read $\forall x \exists y \forall z ((r(x,y) \wedge s(y,z)) \Rightarrow (\neg s(a,y) \vee r(x,z)))$
  - or check that it is equivalent to $\forall x \exists y \forall z (r(x,z) \vee \neg r(x,y) \vee \neg s(y,z) \vee \neg s(a,y))$
- Also not-so-easy for machines (only semi-decidable)

Are there better suited alternatives?

- Can we help users learn/speak/interact with logic?
- Or perhaps not use logic at all?

Early KR Formalisms

Were mostly graphical because pictures are:

- easier to grasp:

  "A picture says more than a thousand words."

- close to how knowledge is represented in human beings (?)

Early KR Formalisms

Were mostly graphical because pictures are:

- easier to grasp:

  "A picture says more than a thousand words."

- close to how knowledge is represented in human beings (?)

Most graphical KR formalisms represent knowledge as graphs with

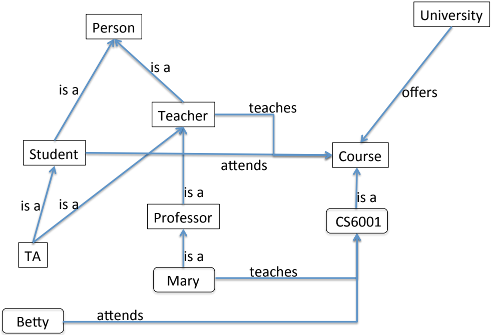
- labeled nodes

  mostly representing concepts, classes, individuals etc.

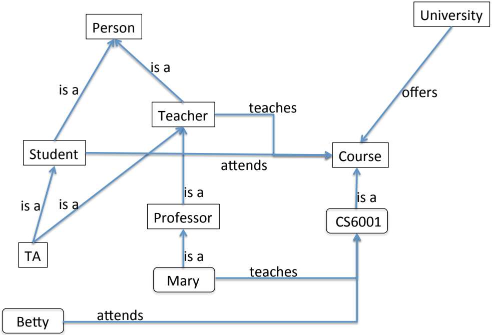
- labeled edges

  mostly representing properties, relationships etc.

Semantic Networks

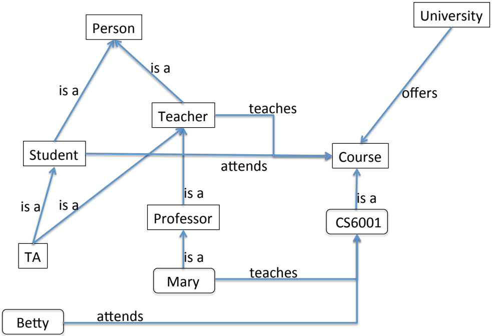


Semantic Networks



What does this graph say exactly? Is Betty a Student?

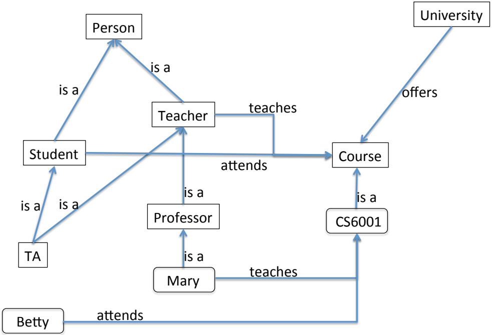
Semantic Networks



What does this graph say exactly? Is Betty a Student?

Problem: it is unclear. Semantics is missing or implicit.

Semantic Networks



What does this graph say exactly? Is Betty a Student?

Problem: it is unclear. Semantics is missing or implicit.

Remedy: base your picture on logic or use logic directly

Simplicty of Semantic Networks is Problematic

1975, What's in a Link (Woods):

"I think we must begin with the realization that there is currently no theory of semantic networks."

Simplicty of Semantic Networks is Problematic

1975, What's in a Link (Woods):

"I think we must begin with the realization that there is currently no theory of semantic networks."

1983, What IS-A is and isn't (Brachman):

"There are almost as many meanings for the IS-A link as there are knowledge representation systems."

17 distinct interpretations of IS-A!

Simplicty of Semantic Networks is Problematic

1975, What's in a Link (Woods):

"I think we must begin with the realization that there is currently no theory of semantic networks."

1983, What IS-A is and isn't (Brachman):

"There are almost as many meanings for the IS-A link as there are knowledge representation systems."

17 distinct interpretations of IS-A!

Instead we need a KR formalism which

- has a well-defined semantics
- is reasonably accessible to users
- balances expressivity and computational practicality

Simplicty of Semantic Networks is Problematic

1975, What's in a Link (Woods):

"I think we must begin with the realization that there is currently no theory of semantic networks."

1983, What IS-A is and isn't (Brachman):

"There are almost as many meanings for the IS-A link as there are knowledge representation systems."

17 distinct interpretations of IS-A!

Instead we need a KR formalism which

- has a well-defined semantics
- is reasonably accessible to users
- balances expressivity and computational practicality

DLs came out of the effort to design such formalism

Terminological Knowledge and Facts

DLs: designed to represent terminological or conceptual knowledge

Terminological Knowledge and Facts

DLs: designed to represent terminological or conceptual knowledge

Goals:

- Formalise basic terminology of an application domain;
- Enable reasoning about concepts:
  - Can there be Mammals?
  - Is every Mammal an Animal?
  - Are Frogs Reptiles?

Terminological Knowledge and Facts

DLs: designed to represent terminological or conceptual knowledge

Goals:

- Formalise basic terminology of an application domain;
- Enable reasoning about concepts:
  - Can there be Mammals?
  - Is every Mammal an Animal?
  - Are Frogs Reptiles?
- Represent facts about individuals
- Enable reasoning about individuals and concepts:
  - Are my facts consistent with my terminology?
  - Is *Kermit* a Frog?

Reasoning

By "reasoning" we understand deductive inference

- From general knowledge to specific conclusions
- All results are necessarily true
  If α follows from Σ, then $\neg\alpha$ is inconsistent with Σ

Reasoning

By “reasoning” we understand deductive inference

- From general knowledge to specific conclusions
- All results are necessarily true
  If α follows from Σ, then $\neg\alpha$ is inconsistent with Σ

Σ : *ESSLLI attendants are students. John attends ESSLLI.*

α : *John is a student*

Reasoning

By "reasoning" we understand deductive inference

- From general knowledge to specific conclusions
- All results are necessarily true
  If α follows from Σ, then $\neg\alpha$ is inconsistent with Σ

Σ : *ESSLLI attendants are students. John attends ESSLLI.*

α : *John is a student*

We never induce relationships from examples

John attends ESSLLI, Mary attends ESSLLI, Jim attends ESSLLI, Maria attends ESSLLI,...

α : *ESSLLI attendants are students*

Applications of Description Logics

Medical Informatics

Applications of Description Logics

Medical Informatics

- SNOMED CT

  Systematized Nomenclature of Medicine – Clinical Terms
  clinical terminology (used for EHR, clinical DSS, etc.)
  $>$300,000 classes (diseases, conditions, etc.)

- NCI Thesaurus (NCI = National Cancer Institute of the USA)

  vocabulary for clinical care, translational and basic research, public information, administrative activities
  Information on $>$10,000 cancers

- ICD 11 (International Classification of Diseases)

  used worldwide for health statistics
  when someone dies, there's always a code from ICD 11

Applications of Description Logics

Bioinformatics

- GO (Gene Ontology)

  controlled vocabulary of terms for gene product characteristics and gene product annotation data

- Bioportal

  REST/Web UI access to 255 bio-health ontologies

Applications of Description Logics

Bioinformatics

- GO (Gene Ontology)

  controlled vocabulary of terms for gene product characteristics and gene product annotation data

- Bioportal

  REST/Web UI access to 255 bio-health ontologies

Semantic Web

- Supply meaning to (linked open) data
- Use TBox when querying data (Lecture 5 will cover this)
  - ontology-based data access
  - data intergration

Why Reasoning is Important?

Helps to avoid or fix errors during ontology development or use:

Why Reasoning is Important?

Helps to avoid or fix errors during ontology development or use:

- Inconsistencies: “protein P1 is located in L1, protein P2 is located in L2 disjoint with L1, interaction I was recorded between P1 and P2”

Why Reasoning is Important?

Helps to avoid or fix errors during ontology development or use:

- Inconsistencies: “protein P1 is located in L1, protein P2 is located in L2 disjoint with L1, interaction I was recorded between P1 and P2”
- Wrong conclusions:
  “Flu is inferred to be a sub-concept of Cancer”

Why Reasoning is Important?

Helps to avoid or fix errors during ontology development or use:

- Inconsistencies: "protein P1 is located in L1, protein P2 is located in L2 disjoint with L1, interaction I was recorded between P1 and P2"
- Wrong conclusions:
  "Flu is inferred to be a sub-concept of Cancer"
- Missing conclusions:
  "Flu is not inferred to be a sub-concept of ViralDisease"

Why Reasoning is Important?

Helps to avoid or fix errors during ontology development or use:

- Inconsistencies: "protein P1 is located in L1, protein P2 is located in L2 disjoint with L1, interaction I was recorded between P1 and P2"
- Wrong conclusions:
  "Flu is inferred to be a sub-concept of Cancer"
- Missing conclusions:
  "Flu is not inferred to be a sub-concept of ViralDisease"

Reasoning enables definition-oriented development

- User does not assert relations, only writes definitions
- Reasoning infers the concept hierarchy

Why Reasoning is Important?

Helps to avoid or fix errors during ontology development or use:

- Inconsistencies: "protein P1 is located in L1, protein P2 is located in L2 disjoint with L1, interaction I was recorded between P1 and P2"
- Wrong conclusions:
  "Flu is inferred to be a sub-concept of Cancer"
- Missing conclusions:
  "Flu is not inferred to be a sub-concept of ViralDisease"

Reasoning enables definition-oriented development

- User does not assert relations, only writes definitions
- Reasoning infers the concept hierarchy

Reasoning can be used to explain the errors

Origins of Description Logics

Basics: Syntax, Semantics, Reasoning Problems

Anatomy of a Reasoner

Syntax and Semantics

What you should remember from your logic class!

Any logic has two key components:

- Syntax: formal language used to write formulas of the logic
- Semantics: specifies how to interpret those formulas

Syntax and Semantics

What you should remember from your logic class!

Any logic has two key components:

- Syntax: formal language used to write formulas of the logic
- Semantics: specifies how to interpret those formulas

Why?

- Syntax: machines can parse/reject formulas
  specified using a formal grammar (EBNF, etc)
- Semantics: machines can understand them
  specified using the language of the Set Theory
- Programming language analogy:
  Syntax and semantics specified in the Standard (e.g., C++)

Concept Language

Core part of a DL: its concept language, e.g.:

$$\mathsf{Animal} \sqcap \exists \mathsf{hasPart}.\mathsf{Feather}$$

describes all animals that are related via hasPart to a feather

Concept Language

Core part of a DL: its concept language, e.g.:

$$\mathsf{Animal} \sqcap \exists \mathsf{hasPart}.\mathsf{Feather}$$

describes all animals that are related via hasPart to a feather

Syntactic components of a concept language:

- Concept names: stand for sets of elements, e.g., Animal
- Role names: stand for binary relations between elements, e.g., hasPart
- Constructors: used to build concept expressions: $\sqcap, \sqcup, \exists, \forall$

Syntax of $\mathcal{ALC}$

$\mathcal{ALC}$ is one of the basic/earliest description logics

- properly contains propositional logic
- enough expressivity for conceptual graphs
- notational variant of well-studied modal logic K_N

Syntax of $\mathcal{ALC}$

$\mathcal{ALC}$ is one of the basic/earliest description logics

- properly contains propositional logic
- enough expressivity for conceptual graphs
- notational variant of well-studied modal logic K_N

The set of concepts in $\mathcal{ALC}$ is defined recursively as follows:

- Every concept name is a concept
- $\top, \bot$ are concepts (pronounced "top" and "bottom")
- $C \sqcap D$, $C \sqcup D$, and $\neg C$ are concepts if C and D are
- Role restrictions are concepts if C is a concept and R is a role

  $\exists \mathsf{R}.C$ existential restriction

  $\forall \mathsf{R}.C$ universal restriction

Semantics of $\mathcal{ALC}$

Semantics given via interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where:

- $\Delta^{\mathcal{I}}$ is a non-empty set (the domain),
- $\cdot^{\mathcal{I}}$ is a mapping (the interpretation function)

Semantics of $\mathcal{ALC}$

Semantics given via interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where:

- $\Delta^\mathcal{I}$ is a non-empty set (the domain),
- $\cdot^\mathcal{I}$ is a mapping (the interpretation function), defined as:

Constructor	Syntax	Example	Semantics
concept name	A	Human	$\mathsf{A}^\mathcal{I} \subseteq \Delta^\mathcal{I}$
role name	R	likes	$\mathsf{R}^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
Top concept	$\top$		$\top^\mathcal{I} \equiv \Delta^\mathcal{I}$
Bottom concept	$\bot$		$\bot^\mathcal{I} \equiv \emptyset$

Semantics of $\mathcal{ALC}$

Semantics given via interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where:

- $\Delta^{\mathcal{I}}$ is a non-empty set (the domain),
- $\cdot^{\mathcal{I}}$ is a mapping (the interpretation function), defined as:

Constructor	Syntax	Example	Semantics
concept name	A	Human	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	R	likes	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Top concept	$\top$		$\top^{\mathcal{I}} \equiv \Delta^{\mathcal{I}}$
Bottom concept	$\bot$		$\bot^{\mathcal{I}} \equiv \emptyset$
conjunction	$C \sqcap D$	Human $\sqcap$ Male	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	Nice $\sqcup$ Rich	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\neg$Meat	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
restrictions:			
existential	$\exists \mathsf{R}.C$	$\exists$hasChild.Human	$\{x \mid \exists y.(x,y) \in \mathsf{R}^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal	$\forall \mathsf{R}.C$	$\forall$hasChild.Blond	$\{x \mid \forall y.(x,y) \in \mathsf{R}^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$

The Griffin Family

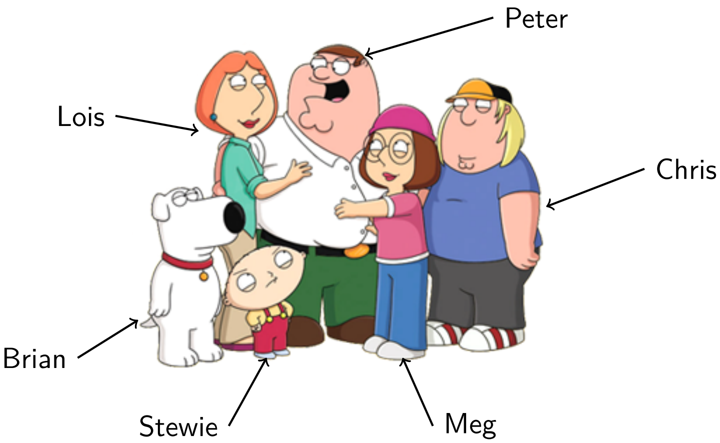
Look at interpretations on some real example

The Griffin Family

Look at interpretations on some real example

The Griffin Family

Look at interpretations on some real example



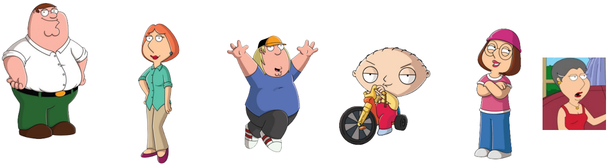
The Griffin Family

We pick some $\mathcal{I}$. It fixes the interpretation of base terms:

The Griffin Family

We pick some $\mathcal{I}$. It fixes the interpretation of base terms:

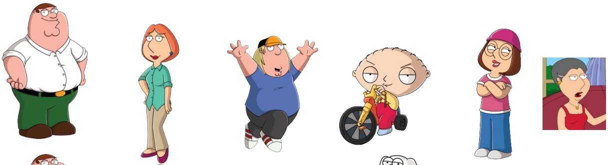
Human$^{\mathcal{I}}$:



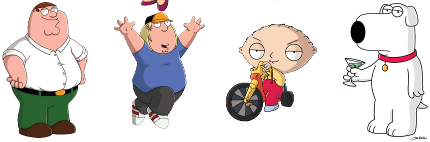
The Griffin Family

We pick some $\mathcal{I}$. It fixes the interpretation of base terms:

$\mathsf{Human}^{\mathcal{I}}$:



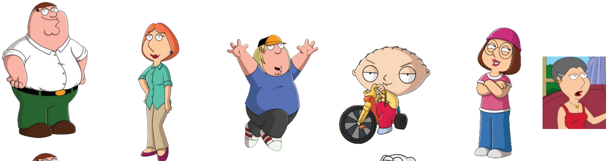
$\mathsf{Male}^{\mathcal{I}}$:



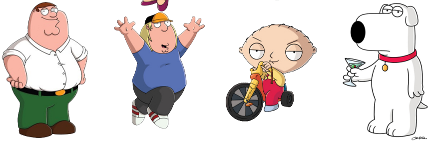
The Griffin Family

We pick some $\mathcal{I}$. It fixes the interpretation of base terms:

$\mathsf{Human}^{\mathcal{I}}$:



$\mathsf{Male}^{\mathcal{I}}$:

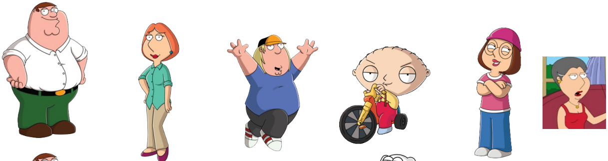


$\mathsf{Parent}^{\mathcal{I}}$:

The Griffin Family

We pick some $\mathcal{I}$. It fixes the interpretation of base terms:

Human$^{\mathcal{I}}$:



Male$^{\mathcal{I}}$:

Parent$^{\mathcal{I}}$:

hasChild$^{\mathcal{I}}$: {(Barbara,Lois), (Peter, Chris), (Peter, Meg),
(Peter, Stewie), (Lois, Chris), (Lois, Meg), (Lois, Stewie)}

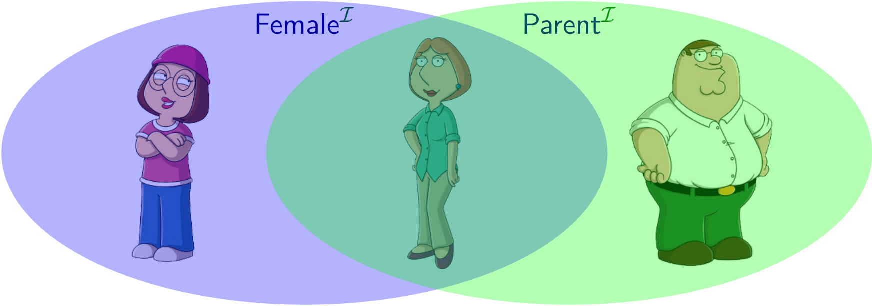
Boolean Connectives Are Easy

$\mathcal{ALC}$ is a propositionally complete language



Boolean Connectives Are Easy

$\mathcal{ALC}$ is a propositionally complete language

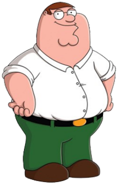


Lois is an instance of $(\mathsf{Female} \sqcap \mathsf{Parent})^{\mathcal{I}}$

Existential Restrictions

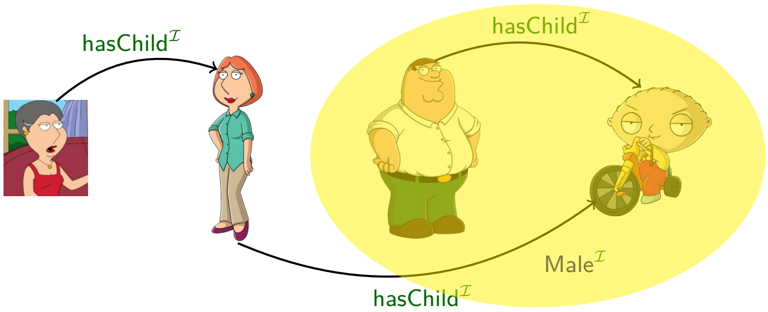
$\exists$hasChild.Male means the set of those who are in hasChild$^{\mathcal{I}}$ relation with an instance of Male$^{\mathcal{I}}$





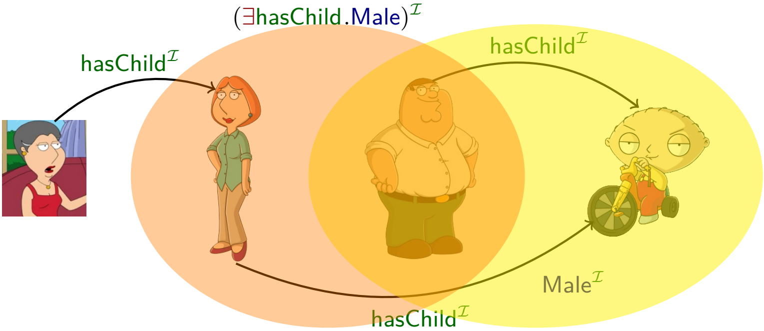

Existential Restrictions

$\exists$hasChild.Male means the set of those who are in $\text{hasChild}^{\mathcal{I}}$ relation with an instance of $\text{Male}^{\mathcal{I}}$



Existential Restrictions

$\exists$hasChild.Male means the set of those who are in hasChild$^{\mathcal{I}}$ relation with an instance of Male$^{\mathcal{I}}$



Universal Restrictions

$\forall$hasChild.Female means the set of those who are in hasChild$^{\mathcal{I}}$ relation with only instances of Female$^{\mathcal{I}}$:
$\{x \mid \forall y.(x, y) \in \text{hasChild}^{\mathcal{I}} \Rightarrow y \in \text{Female}^{\mathcal{I}}\}$

instances of $(\forall\text{hasChild.Female})^{\mathcal{I}}$

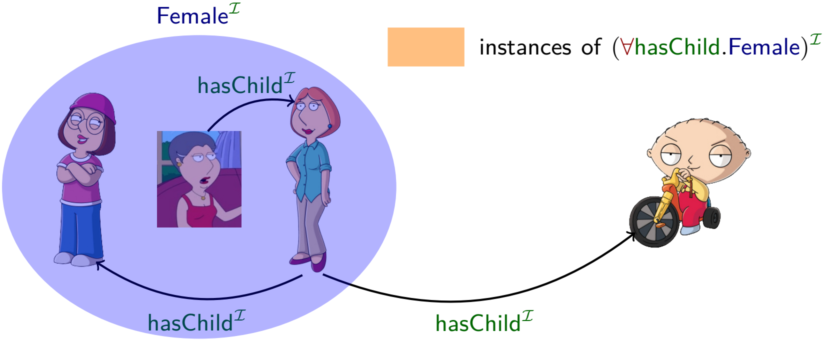



Question: what are the instances of $(\forall\text{hasChild.Female})^{\mathcal{I}}$

Universal Restrictions

$\forall$hasChild.Female means the set of those who are in $\mathsf{hasChild}^{\mathcal{I}}$ relation with only instances of $\mathsf{Female}^{\mathcal{I}}$:

$\{x \mid \forall y.(x, y) \in \mathsf{hasChild}^{\mathcal{I}} \Rightarrow y \in \mathsf{Female}^{\mathcal{I}}\}$

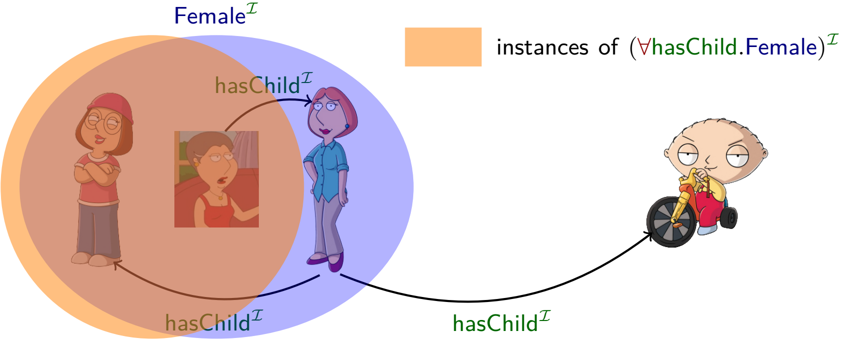


Question: what are the instances of $(\forall\mathsf{hasChild}.\mathsf{Female})^{\mathcal{I}}$

Universal Restrictions

$\forall$hasChild.Female means the set of those who are in hasChild$^{\mathcal{I}}$ relation with only instances of Female$^{\mathcal{I}}$:

$\{x \mid \forall y.(x,y) \in \text{hasChild}^{\mathcal{I}} \Rightarrow y \in \text{Female}^{\mathcal{I}}\}$

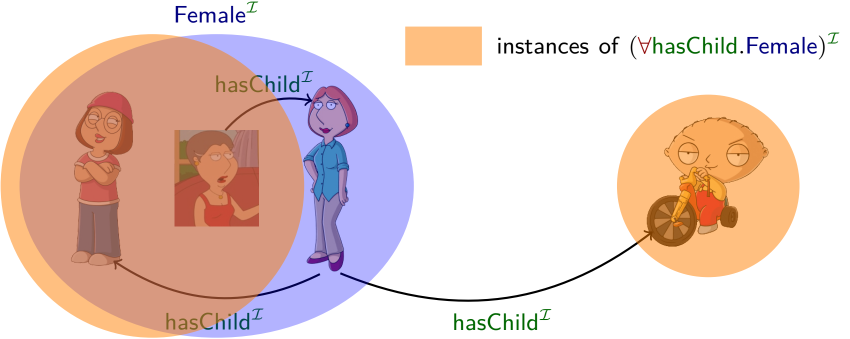


Question: what are the instances of ($\forall$hasChild.Female)$^{\mathcal{I}}$

Universal Restrictions

$\forall$hasChild.Female means the set of those who are in hasChild$^{\mathcal{I}}$ relation with only instances of Female$^{\mathcal{I}}$:

$\{x \mid \forall y.(x, y) \in \text{hasChild}^{\mathcal{I}} \Rightarrow y \in \text{Female}^{\mathcal{I}}\}$



Question: what are the instances of ($\forall$hasChild.Female)$^{\mathcal{I}}$

$\mathcal{ALC}$ Concept Interpretations

$C^{\mathcal{I}}$ can be visualized as a labeled graph $\mathcal{G}_{C^{\mathcal{I}}} = \langle V, E \rangle$, where

- V is a non-empty set of domain elements where $v_0 \in C^{\mathcal{I}}$
- $D \in L(v)$ if $v \in D^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$

$\mathcal{ALC}$ Concept Interpretations

$C^{\mathcal{I}}$ can be visualized as a labeled graph $\mathcal{G}_{C^{\mathcal{I}}} = \langle V, E \rangle$, where

- V is a non-empty set of domain elements where $v_0 \in C^{\mathcal{I}}$
- $D \in L(v)$ if $v \in D^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$

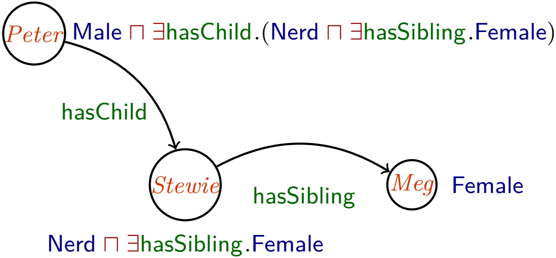
Example: Male $\sqcap$ $\exists$hasChild.(Nerd $\sqcap$ $\exists$hasSibling.Female)

$\mathcal{ALC}$ Concept Interpretations

$C^{\mathcal{I}}$ can be visualized as a labeled graph $\mathcal{G}_{C^{\mathcal{I}}} = \langle V, E \rangle$, where

- V is a non-empty set of domain elements where $v_0 \in C^{\mathcal{I}}$
- $D \in L(v)$ if $v \in D^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$

Example: Male ⊓ ∃hasChild.(Nerd ⊓ ∃hasSibling.Female)



Basic Reasoning Problems

Definition: let C, D be $\mathcal{ALC}$ concepts. We say that

- C is satisfiable if there exists some $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.
- C is subsumed by D (written $\emptyset \models C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$) if for every interpretation $\mathcal{I}$, it is true that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Basic Reasoning Problems

Definition: let C, D be $\mathcal{ALC}$ concepts. We say that

- C is satisfiable if there exists some $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.
- C is subsumed by D (written $\emptyset \models C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$) if for every interpretation $\mathcal{I}$, it is true that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Question: Which of the following concepts is satisfiable? Which is subsumed by which?

(1) $\exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B})$ (2) $\exists \mathsf{R}.(\mathsf{A} \sqcup \mathsf{B})$

(3) $\forall \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B})$ (4) $\exists \mathsf{R}.(\mathsf{A} \sqcap \neg \mathsf{A})$

(5) $\exists \mathsf{R}.\mathsf{A} \sqcap \forall \mathsf{R}.\mathsf{B}$ (6) $\exists \mathsf{R}.\mathsf{A}$

(7) $\exists \mathsf{R}.\mathsf{A} \sqcap \forall \mathsf{R}.\neg \mathsf{A}$ (8) $\exists \mathsf{R}.\mathsf{A} \sqcap \forall \mathsf{S}.\neg \mathsf{A}$

The TBox (Terminology)

Definition

- A general concept inclusion (GCI) is a statement of the form $C \sqsubseteq D$, where C, D are (possibly complex) concepts
- A (general) TBox is a finite set of GCIs: $\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leqslant i \leqslant n\}$

The TBox (Terminology)

Definition

- A general concept inclusion (GCI) is a statement of the form $C \sqsubseteq D$, where C, D are (possibly complex) concepts
- A (general) TBox is a finite set of GCIs: $\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leqslant i \leqslant n\}$
- $\mathcal{I}$ satisfies $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (written $\mathcal{I} \models C \sqsubseteq D$)
- $\mathcal{I}$ is a model of TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies every $C_i \sqsubseteq D_i$
- We use $C \equiv D$ to abbreviate $C \sqsubseteq D$, $D \sqsubseteq C$

The TBox (Terminology)

Definition

- A general concept inclusion (GCI) is a statement of the form $C \sqsubseteq D$, where C, D are (possibly complex) concepts
- A (general) TBox is a finite set of GCIs: $\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leqslant i \leqslant n\}$
- $\mathcal{I}$ satisfies $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (written $\mathcal{I} \models C \sqsubseteq D$)
- $\mathcal{I}$ is a model of TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies every $C_i \sqsubseteq D_i$
- We use $C \equiv D$ to abbreviate $C \sqsubseteq D$, $D \sqsubseteq C$

Example:

$$\begin{aligned} \{ \ \mathsf{Father} &\equiv \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\mathsf{Human}, \\ \mathsf{Human} &\equiv \mathsf{Mammal} \sqcap \forall \mathsf{hasParent}.\mathsf{Human}, \\ \exists \mathsf{favourite}.\mathsf{Brewery} &\sqsubseteq \exists \mathsf{drinks}.\mathsf{Beer} \ \} \end{aligned}$$

$\mathcal{ALC}$ TBox Interpretations

$\mathcal{T}^{\mathcal{I}}$ can be visualized just as $C^{\mathcal{I}}$, as $\mathcal{G}_{\mathcal{T}^{\mathcal{I}}} = \langle V, E \rangle$, where

- $D \in L(v)$ if $v \in D^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$
- one may start with one node for each concept name in $\mathcal{T}$

$\mathcal{ALC}$ TBox Interpretations

$\mathcal{T}^\mathcal{I}$ can be visualized just as $C^\mathcal{I}$, as $\mathcal{G}_{\mathcal{T}^\mathcal{I}} = \langle V, E \rangle$, where

- $D \in L(v)$ if $v \in D^\mathcal{I}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^\mathcal{I}$
- one may start with one node for each concept name in $\mathcal{T}$

Example:

$$\begin{aligned}\{\ \mathsf{Father} &\equiv \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\mathsf{Human}\,,\\ \mathsf{Human} &\equiv \mathsf{Mammal} \sqcap \forall \mathsf{hasParent}.\mathsf{Human}\,,\\ \exists \mathsf{favourite}.\mathsf{Brewery} &\sqsubseteq \exists \mathsf{drinks}.\mathsf{Beer}\ \}\end{aligned}$$

Exercise:

1. Draw one model of this TBox
2. Draw one non-model of this TBox

Reasoning Problems w.r.t. TBox

Definition: let C, D be concepts, $\mathcal{T}$ a TBox. We say that

- C is satisfiable w.r.t. $\mathcal{T}$
  if there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$
- C is subsumed by D w.r.t. $\mathcal{T}$ (written $\mathcal{T} \models C \sqsubseteq D$)
  if, for every model $\mathcal{I}$ of $\mathcal{T}$, we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Reasoning Problems w.r.t. TBox

Definition: let C, D be concepts, $\mathcal{T}$ a TBox. We say that

- C is satisfiable w.r.t. $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$
- C is subsumed by D w.r.t. $\mathcal{T}$ (written $\mathcal{T} \models C \sqsubseteq D$) if, for every model $\mathcal{I}$ of $\mathcal{T}$, we have $C^\mathcal{I} \subseteq D^\mathcal{I}$

Example:

$$\mathcal{T} = \{ \begin{array}{rcl} \mathsf{A} & \sqsubseteq & \mathsf{B} \sqcap \exists \mathsf{R}.\mathsf{C}, \\ \exists \mathsf{R}.\top & \sqsubseteq & \neg \mathsf{A} \end{array} \}$$

Questions: Does $\mathcal{T}$ have a model?

Reasoning Problems w.r.t. TBox

Definition: let C, D be concepts, $\mathcal{T}$ a TBox. We say that

- C is satisfiable w.r.t. $\mathcal{T}$
  if there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$
- C is subsumed by D w.r.t. $\mathcal{T}$ (written $\mathcal{T} \models C \sqsubseteq D$)
  if, for every model $\mathcal{I}$ of $\mathcal{T}$, we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Example:

$$\mathcal{T} = \{ \begin{array}{rcl} \mathsf{A} & \sqsubseteq & \mathsf{B} \sqcap \exists \mathsf{R}.\mathsf{C}, \\ \exists \mathsf{R}.\top & \sqsubseteq & \neg \mathsf{A} \end{array} \}$$

Questions: Does $\mathcal{T}$ have a model?
Are all concept names in $\mathcal{T}$ satisfiable?

TBox + ABox $\equiv$ Ontology

TBox

- captures knowledge on a general, conceptual level
- contains concept def.s + general axioms about concepts

TBox + ABox $\equiv$ Ontology

TBox
- captures knowledge on a general, conceptual level
- contains concept def.s + general axioms about concepts

Think databases: TBox defines schema. Where's data?

TBox + ABox ≡ Ontology

TBox
- captures knowledge on a general, conceptual level
- contains concept def.s + general axioms about concepts

Think databases: TBox defines schema. Where's data?

ABox
- captures knowledge on an individual level
- is a finite set of
  - concept assertions $a : C$ e.g., $John$: Man,
  - role assertions (a, b) : R e.g., $(John, Mary)$: hasChild
- uses terms (concepts, roles) defined in the TBox

TBox + ABox $\equiv$ Ontology

TBox

- captures knowledge on a general, conceptual level
- contains concept def.s + general axioms about concepts

Think databases: TBox defines schema. Where's data?

ABox

- captures knowledge on an individual level
- is a finite set of
  - concept assertions $a : C$ e.g., $John$: Man,
  - role assertions (a, b) : R e.g., $(John, Mary)$: hasChild
- uses terms (concepts, roles) defined in the TBox

A pair of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ is called ontology

$\mathcal{O} = (\mathcal{T}, \mathcal{A})$

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- satisfies a role assertion $(a, b) : \mathsf{R}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- satisfies a role assertion $(a, b) : \mathsf{R}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$
- is a model of an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies each assertion in $\mathcal{A}$

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- satisfies a role assertion $(a, b) : \mathsf{R}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$
- is a model of an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies each assertion in $\mathcal{A}$

$a : C$ is entailed by $\mathcal{A}$ if every model of $\mathcal{A}$ satisfies $a : C$

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- satisfies a role assertion $(a, b) : \mathsf{R}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$
- is a model of an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies each assertion in $\mathcal{A}$

$a : C$ is entailed by $\mathcal{A}$ if every model of $\mathcal{A}$ satisfies $a : C$

Question: why do I not define entailments of role assertions?

$\mathcal{ALC}$ ABox Interpretations

Semantics: an interpretation $\mathcal{I}$

- maps each individual name e to some $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- satisfies a role assertion $(a, b) : \mathsf{R}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$
- is a model of an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies each assertion in $\mathcal{A}$

$a : C$ is entailed by $\mathcal{A}$ if every model of $\mathcal{A}$ satisfies $a : C$

Question: why do I not define entailments of role assertions?

Answer: no non-trivial role assertion entailments in $\mathcal{ALC}$! $\ddot\smile$

$\mathcal{ALC}$ ABox Interpretations

$\mathcal{A}^{\mathcal{I}}$ can be visualized similarly to $\mathcal{T}^{\mathcal{I}}$, as $\mathcal{G}_{\mathcal{A}^{\mathcal{I}}} = \langle V, E \rangle$, where

- $C \in L(v)$ if $v \in C^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$
- one may start with one node for each individual name in $\mathcal{A}$

$\mathcal{ALC}$ ABox Interpretations

$\mathcal{A}^{\mathcal{I}}$ can be visualized similarly to $\mathcal{T}^{\mathcal{I}}$, as $\mathcal{G}_{\mathcal{A}^{\mathcal{I}}} = \langle V, E \rangle$, where

- $C \in L(v)$ if $v \in C^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$
- one may start with one node for each individual name in $\mathcal{A}$

Example:

$$\mathcal{A} = \{ \quad a : \mathsf{B} \sqcap \exists \mathsf{R}.\mathsf{C}, \quad b : \mathsf{A} \sqcap \neg \mathsf{D} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}, \quad (b, a) : \mathsf{S} \ \}$$

Question: can you see any entailments?

Ontology Semantics

Combined interpretation for TBox and ABox

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model
- $\mathcal{O}$ is coherent if all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model
- $\mathcal{O}$ is coherent if all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$
- $C \sqsubseteq D$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model
- $\mathcal{O}$ is coherent if all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$
- $C \sqsubseteq D$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $a : C$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $a^{\mathcal{I}} \in C^{\mathcal{I}}$

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model
- $\mathcal{O}$ is coherent if all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$
- $C \sqsubseteq D$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $a : C$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $a^{\mathcal{I}} \in C^{\mathcal{I}}$

Example:

$$\mathcal{O} = \{ \quad \mathsf{A} \sqsubseteq \mathsf{B} \sqcap \exists \mathsf{R}.\mathsf{C}, \qquad a : \mathsf{B}, \\ \exists \mathsf{R}.\top \sqsubseteq \neg \mathsf{A}, \qquad (a, b) : \mathsf{R} \quad \}$$

Questions: Does $\mathcal{O}$ have a model?
Can you see any entailments?
What about $\mathcal{O} \cup \{b : \mathsf{A}\}$?

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model
- $\mathcal{O}$ is coherent if all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$
- $C \sqsubseteq D$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $a : C$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $a^{\mathcal{I}} \in C^{\mathcal{I}}$

Lemma: $C \sqsubseteq D$ is entailed by $(\mathcal{T}, \mathcal{A})$ iff $C \sqsubseteq D$ is entailed by $\mathcal{T}$

Ontology Semantics

Combined interpretation for TBox and ABox

- Int. $\mathcal{I}$ is a model of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$
- $\mathcal{O}$ is consistent if it has some model
- $\mathcal{O}$ is coherent if all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$
- $C \sqsubseteq D$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $a : C$ is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ satisfies $a^{\mathcal{I}} \in C^{\mathcal{I}}$

Lemma: $C \sqsubseteq D$ is entailed by $(\mathcal{T}, \mathcal{A})$ iff $C \sqsubseteq D$ is entailed by $\mathcal{T}$

This has big practical impact for reasoners

- Schema is often small
- Data is often large

Description Logics and OWL

OWL is W3C-standardized Web Ontology Language

- If you publish your ontology, it should be in OWL
- If you don't, then it better be in OWL

Description Logics and OWL

OWL is W3C-standardized Web Ontology Language

- If you publish your ontology, it should be in OWL
- If you don't, then it better be in OWL

OWL is basically a DL + a common syntax

- Syntax: OWL/XML, Functional, Manchester, RDF-based
- Semantics: DL model theory
  any reasoning in OWL is reduced to reasoning in DL

Description Logics and OWL

OWL is W3C-standardized Web Ontology Language

- If you publish your ontology, it should be in OWL
- If you don't, then it better be in OWL

OWL is basically a DL + a common syntax

- Syntax: OWL/XML, Functional, Manchester, RDF-based
- Semantics: DL model theory
  any reasoning in OWL is reduced to reasoning in DL

OWL includes more stuff:

- Datatypes: strings, integers, dates, etc.
  a.k.a. concrete domains in some DLs
- Non-logical stuff: annotations

OWL Profiles

OWL is a family of languages designed for specific scenarios

OWL Profiles

OWL is a family of languages designed for specific scenarios

Profile	DL	Scenario
OWL EL	$\mathcal{EL}^{++}$	Maintaining large but simple terminologies (for HCLS, biology, etc., Lecture 4)

OWL Profiles

OWL is a family of languages designed for specific scenarios

Profile	DL	Scenario
OWL EL	$\mathcal{EL}^{++}$	Maintaining large but simple terminologies (for HCLS, biology, etc., Lecture 4)
OWL QL	DL-Lite	Scalable ontology-based data access (Lecture 5)
OWL RL	DLP	Rule-based applications

OWL Profiles

OWL is a family of languages designed for specific scenarios

Profile	DL	Scenario
OWL EL	$\mathcal{EL}^{++}$	Maintaining large but simple terminologies (for HCLS, biology, etc., Lecture 4)
OWL QL	DL-Lite	Scalable ontology-based data access (Lecture 5)
OWL RL	DLP	Rule-based applications
OWL DL	$\mathcal{SROIQ}$	Encapsulates all above, remains decidable. Reasoning is tableau-based (Lecture 2)

OWL Profiles

OWL is a family of languages designed for specific scenarios

Profile	DL	Scenario
OWL EL	$\mathcal{EL}^{++}$	Maintaining large but simple terminologies (for HCLS, biology, etc., Lecture 4)
OWL QL	DL-Lite	Scalable ontology-based data access (Lecture 5)
OWL RL	DLP	Rule-based applications
OWL DL	$\mathcal{SROIQ}$	Encapsulates all above, remains decidable. Reasoning is tableau-based (Lecture 2)

Each profile trades some expressivity for computational guarantees:

- OWL EL: PTime classification
- OWL QL: scalable query answering
- OWL RL: completeness w.r.t. rule systems

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)
- classification $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$?
  (for all concept names A, B)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)
- classification $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$?
  (for all concept names A, B)
- realization $\mathcal{O} \models b \colon \mathsf{B}$?
  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)
- classification $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$?
  (for all concept names A, B)
- realization $\mathcal{O} \models b\colon \mathsf{B}$?
  (for all concept names A, individual names b)

Question: do we need 4 different algorithms for these?

Consistency Suffices

Theorem: Let $\mathcal{O}$ be an ontology and a a fresh individual. Then:

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{a\colon C\}$ is consistent

Consistency Suffices

Theorem: Let $\mathcal{O}$ be an ontology and a a fresh individual. Then:

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{a\colon C\}$ is consistent
2. $\mathcal{O}$ is coherent iff $\mathcal{O} \cup \{a\colon \mathsf{A}\}$ is consistent
   (for each concept name A)

Consistency Suffices

Theorem: Let $\mathcal{O}$ be an ontology and a a fresh individual. Then:

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{a\colon C\}$ is consistent
2. $\mathcal{O}$ is coherent iff $\mathcal{O} \cup \{a\colon \mathsf{A}\}$ is consistent (for each concept name A)
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{a\colon (C \sqcap \neg D)\}$ is not consistent

Consistency Suffices

Theorem: Let $\mathcal{O}$ be an ontology and a a fresh individual. Then:

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{a \colon C\}$ is consistent
2. $\mathcal{O}$ is coherent iff $\mathcal{O} \cup \{a \colon \mathsf{A}\}$ is consistent
   (for each concept name A)
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{a \colon (C \sqcap \neg D)\}$ is not consistent
4. $\mathcal{O} \models b \colon C$ iff $\mathcal{O} \cup \{b \colon \neg C\}$ is not consistent

Answer: a decision procedure to solve consistency decides all standard DL reasoning problems

Consistency Suffices

Theorem: Let $\mathcal{O}$ be an ontology and a a fresh individual. Then:

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{a\colon C\}$ is consistent
2. $\mathcal{O}$ is coherent iff $\mathcal{O} \cup \{a\colon \mathsf{A}\}$ is consistent (for each concept name A)
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{a\colon (C \sqcap \neg D)\}$ is not consistent
4. $\mathcal{O} \models b\colon C$ iff $\mathcal{O} \cup \{b\colon \neg C\}$ is not consistent

Answer: a decision procedure to solve consistency decides all standard DL reasoning problems

This does not mean that the naive reduction is practical!

Origins of Description Logics

Basics: Syntax, Semantics, Reasoning Problems

Anatomy of a Reasoner

What is Reasoner

Reasoner is a system that solves DL reasoning problems

What is Reasoner

Reasoner is a system that solves DL reasoning problems

- Input: ontology (+ an axiom, e.g., $C \sqsubseteq D$)
- Output: yes/no (consistency, entailment), concept hierarchy (classification)

  yes/no for consistency, entailment, satisfiability
  concept hierarchy for classification
  individual to concepts mapping for realization

What is Reasoner

Reasoner is a system that solves DL reasoning problems

- Input: ontology (+ an axiom, e.g., $C \sqsubseteq D$)
- Output: yes/no (consistency, entailment), concept hierarchy (classification)

  yes/no for consistency, entailment, satisfiability
  concept hierarchy for classification
  individual to concepts mapping for realization

Reasoner is more than just implementations of algorithms

- It has to interact with the world (suitable APIs, load data,...)
- It has to convert the input into a suitable form
- It has to invoke the right algorithm at the right time
- It has to manage optimizations

Reasoner: the Main Layers

Interface Layer

Parsing | API bindings | Protégé plugin

Internal Data Model

(Terms, $\sqcap, \sqcup, \exists, \forall$)

Intermediate Layer

Pre-processing, indexing, caching, taxonomy
Incomplete reasoning

Reasoning Layer

Classification
Realization

Core reasoning procedure

Query answering

Interface Layer

Reasoner must be able to interact with world

- Load ontologies and export the results
- Interact with software via standard APIs (OWL API)
- Be useable in ontology editors (Protégé)

Interface Layer

Reasoner must be able to interact with world

- Load ontologies and export the results
- Interact with software via standard APIs (OWL API)
- Be useable in ontology editors (Protégé)

Commonly provided functionality:

- Implementation of standard interfaces (`OWLReasoner` in OWL API)
- Parsers
- Serializers

Interface Layer

Reasoner must be able to interact with world

- Load ontologies and export the results
- Interact with software via standard APIs (OWL API)
- Be useable in ontology editors (Protégé)

Commonly provided functionality:

- Implementation of standard interfaces (`OWLReasoner` in OWL API)
- Parsers
- Serializers

APIs usually provide parsers, serializers, and the data model. . .

. . . but reasoners often support their own for efficiency

Internal Data Model

Internal Data Model is representation of loaded knowledge

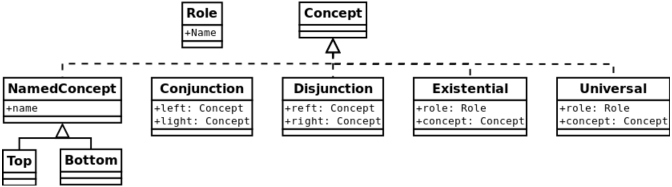
- Optimized for reasoning tasks
- Covers supported features of the language

Internal Data Model

Internal Data Model is representation of loaded knowledge

- Optimized for reasoning tasks
- Covers supported features of the language

Example: Object-Oriented API for $\mathcal{ALC}$

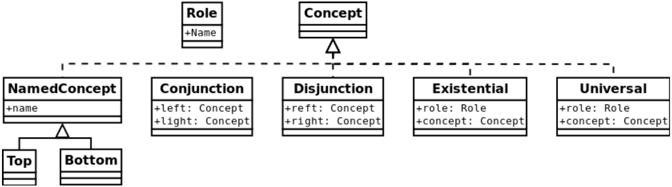


Internal Data Model

Internal Data Model is representation of loaded knowledge

- Optimized for reasoning tasks
- Covers supported features of the language

Example: Object-Oriented API for $\mathcal{ALC}$



IDM does not have to mirror the language model

- Can opt for the minimal sufficient set of constructors
- ELK does not store axioms, only rules

Intermediate Layer: Pre-processing and Indexing

Pre-processing: massaging data before sending to reasoning layer

Purely syntactic axioms/concept rewriting:

- Normalization: $\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D$
- Simplification: $\exists \mathsf{R}.\mathsf{A} \sqcap \exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B}) \rightsquigarrow \exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B})$
- Absorption: $A \sqcap C \sqsubseteq D \rightsquigarrow \mathsf{A} \sqsubseteq \neg C \sqcup D$

Intermediate Layer: Pre-processing and Indexing

Pre-processing: massaging data before sending to reasoning layer

Purely syntactic axioms/concept rewriting:

- Normalization: $\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D$
- Simplification: $\exists \mathsf{R}.\mathsf{A} \sqcap \exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B}) \rightsquigarrow \exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B})$
- Absorption: $A \sqcap C \sqsubseteq D \rightsquigarrow \mathsf{A} \sqsubseteq \neg C \sqcup D$

Indexing: extra data structure for faster look-ups

- $\mathsf{A} \mapsto$ set of told subsumers
- $\mathsf{A} \mapsto$ set of told disjoint concepts
- ...

Intermediate Layer: Pre-processing and Indexing

Pre-processing: massaging data before sending to reasoning layer

Purely syntactic axioms/concept rewriting:

- Normalization: $\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D$
- Simplification: $\exists \mathsf{R}.\mathsf{A} \sqcap \exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B}) \rightsquigarrow \exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B})$
- Absorption: $A \sqcap C \sqsubseteq D \rightsquigarrow \mathsf{A} \sqsubseteq \neg C \sqcup D$

Indexing: extra data structure for faster look-ups

- $\mathsf{A} \mapsto$ set of told subsumers
- $\mathsf{A} \mapsto$ set of told disjoint concepts
- ...

Practical hint: both can be done in parallel with parsing/loading

Intermediate Layer: Reductions

Reasoner reduces the input problem to the one for which the core procedure is optimized

- Tableau: $\mathcal{O} \stackrel{?}{\models} \alpha \rightsquigarrow$ is $\mathcal{O} \cup \{\neg\alpha\}$ consistent?
- Consequence-based algorithms: $\mathcal{O} \stackrel{?}{\models} \neg C \sqcup D \rightsquigarrow \mathcal{O} \stackrel{?}{\models} C \sqsubseteq D$ (CB algorithms compute subsumers in a goal-directed way)

Intermediate Layer: Reductions

Reasoner reduces the input problem to the one for which the core procedure is optimized

- Tableau: $\mathcal{O} \overset{?}{\models} \alpha \rightsquigarrow$ is $\mathcal{O} \cup \{\neg\alpha\}$ consistent?
- Consequence-based algorithms: $\mathcal{O} \overset{?}{\models} \neg C \sqcup D \rightsquigarrow \mathcal{O} \overset{?}{\models} C \sqsubseteq D$
  (CB algorithms compute subsumers in a goal-directed way)

Reasoner can also reduce the problem to one previously solved

- $\mathcal{O} \overset{?}{\models} \neg C \sqcup D \sqsubseteq \bot \rightsquigarrow \mathcal{O} \overset{?}{\models} C \sqsubseteq D$
  if subsumers for C have been computed

Intermediate Layer: Caching

- Single shot reasoning: just answer one query, e.g., $\mathcal{O} \stackrel{?}{\models} C \sqsubseteq D$, and discard everything
- Multiple reasoning: save and re-use intermediate results

Intermediate Layer: Caching

- Single shot reasoning: just answer one query, e.g., $\mathcal{O} \overset{?}{\models} C \sqsubseteq D$, and discard everything
- Multiple reasoning: save and re-use intermediate results

Reasoner infers a lot more than it shows to the user

Example: >24M inferences when classifying SNOMED CT (“only” 300K concepts)

Intermediate Layer: Caching

- Single shot reasoning: just answer one query, e.g., $\mathcal{O} \overset{?}{\models} C \sqsubseteq D$, and discard everything
- Multiple reasoning: save and re-use intermediate results

Reasoner infers a lot more than it shows to the user

Example: >24M inferences when classifying SNOMED CT ("only" 300K concepts)

Other stuff, e.g., complex subsumers, can be re-used later.
This layer decides:

- what to save
- what to discard (w.r.t. which policy)
- how to look things up

Caching

Example: the reasoner inferred $A \sqsubseteq C \sqcap D$
while checking satisfiability of A

Caching

Example: the reasoner inferred $A \sqsubseteq C \sqcap D$
while checking satisfiability of A

Next task: compute subsumers of A. It can immediately:

- ignore concepts disjoint with C or D, if known
- take subsumers of C or D, if known
- cache all named subsumers of $C \sqcap D$, if it makes sense

Caching

Example: the reasoner inferred $A \sqsubseteq C \sqcap D$ while checking satisfiability of A

Next task: compute subsumers of A. It can immediately:

- ignore concepts disjoint with C or D, if known
- take subsumers of C or D, if known
- cache all named subsumers of $C \sqcap D$, if it makes sense

Caches may be cleared when the ontology is changed

. . . or not! Incremental reasoning algorithms exist

Deletions are particularly tricky. Why?

Intermediate Layer: Incomplete Reasoning

Main reasoning algorithms are nearly always expensive

Often there are cheaper ways to get the answer

- looking up in the cache
- by probing instead of searching systematically

Intermediate Layer: Incomplete Reasoning

Main reasoning algorithms are nearly always expensive

Often there are cheaper ways to get the answer

- looking up in the cache
- by probing instead of searching systematically

Examples:

- $\mathcal{EL}$: if $\perp$ does not occur in $\mathcal{O}$, it cannot be inconsistent
- Detecting obvious conflicts, e.g., $\exists \mathsf{R}.\neg C$ and $\forall \mathsf{R}.C$

Intermediate Layer: Incomplete Reasoning

Main reasoning algorithms are nearly always expensive

Often there are cheaper ways to get the answer

- looking up in the cache
- by probing instead of searching systematically

Examples:

- $\mathcal{EL}$: if $\bot$ does not occur in $\mathcal{O}$, it cannot be inconsistent
- Detecting obvious conflicts, e.g., $\exists \mathsf{R}.\neg C$ and $\forall \mathsf{R}.C$

Approximations can help too

If $\mathcal{O}' \subseteq \mathcal{O}$ and $\mathcal{O}' \models \alpha$, then $\mathcal{O} \models \alpha$ (monotonicity)

$\mathcal{O}'$ can fit into a simpler language $\Rightarrow$ easier to reason with

Reasoning Layer: Core Reasoning Procedure

Implementation of the main reasoning algorithm

- Expressive DLs: usually tableau algorithm for consistency (L2)
- Lightweight DLs: rule-based saturation algorithm (L4)

Reasoning Layer: Core Reasoning Procedure

Implementation of the main reasoning algorithm

- Expressive DLs: usually tableau algorithm for consistency (L2)
- Lightweight DLs: rule-based saturation algorithm (L4)

Should be compact and extensible to:

- New language features
- New optimizations

Reasoning Layer: Core Reasoning Procedure

Implementation of the main reasoning algorithm

- Expressive DLs: usually tableau algorithm for consistency (L2)
- Lightweight DLs: rule-based saturation algorithm (L4)

Should be compact and extensible to:

- New language features
- New optimizations

Should be reusable for higher level tasks:

- Explanations and debugging
  (find all reasons why $\mathcal{O} \models C \sqsubseteq \bot$ happens)
- Query answering
- Incremental reasoning

Reasoning Layer: Classification and Realization

Classification: compute $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$ for all concept names in $\mathcal{O}$

Realization: compute $\mathcal{O} \models a\colon \mathsf{A}$ for all individuals in $\mathcal{O}$

Reasoning Layer: Classification and Realization

Classification: compute $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$ for all concept names in $\mathcal{O}$

Realization: compute $\mathcal{O} \models a\colon \mathsf{A}$ for all individuals in $\mathcal{O}$

Can be reduced to polynomial number of consistency problems

- Very inefficient, many more non-subsumptions than subsumptions (L3)
- Can be computed in one pass for deterministic DLs (L4)

Reasoning Layer: Classification and Realization

Classification: compute $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$ for all concept names in $\mathcal{O}$

Realization: compute $\mathcal{O} \models a \colon \mathsf{A}$ for all individuals in $\mathcal{O}$

Can be reduced to polynomial number of consistency problems

- Very inefficient, many more non-subsumptions than subsumptions (L3)
- Can be computed in one pass for deterministic DLs (L4)

Once the ontology is classified, many tasks are easier

Taxonomy Construction

Users want to see only direct subsumptions

- A is directly subsumed by B if $\mathcal{O} \models A \sqsubseteq B$ and
- There is no C s.t. $\mathcal{O} \models A \sqsubseteq C$ and $\mathcal{O} \models C \sqsubseteq B$

Taxonomy Construction

Users want to see only direct subsumptions

- A is directly subsumed by B if $\mathcal{O} \models A \sqsubseteq B$ and
- There is no C s.t. $\mathcal{O} \models A \sqsubseteq C$ and $\mathcal{O} \models C \sqsubseteq B$

Class taxonomy is transitively reduced graph of subsumptions

Non-trivial, the complexity is $O(n^k)$ where $k > 2$

Taxonomy Construction

Users want to see only direct subsumptions

- A is directly subsumed by B if $\mathcal{O} \models A \sqsubseteq B$ and
- There is no C s.t. $\mathcal{O} \models A \sqsubseteq C$ and $\mathcal{O} \models C \sqsubseteq B$

Class taxonomy is transitively reduced graph of subsumptions

Non-trivial, the complexity is $O(n^k)$ where $k > 2$

Transitive reduction algorithms can

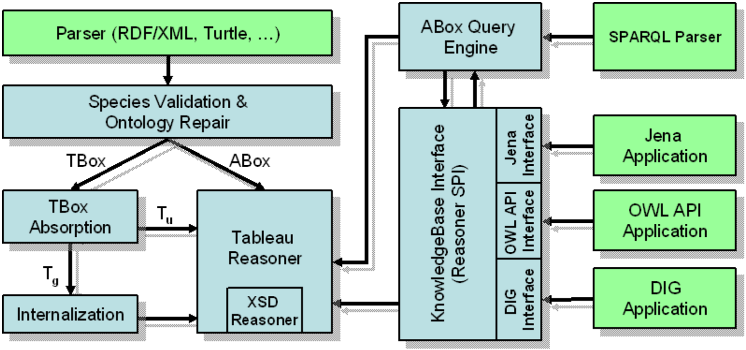
- maintain the reduction as the ontologies is classified
- compute the reduction post factum

Example: Pellet

Pellet – one of the earliest complete reasoners for expressive DLs

Example: Pellet

Pellet – one of the earliest complete reasoners for expressive DLs

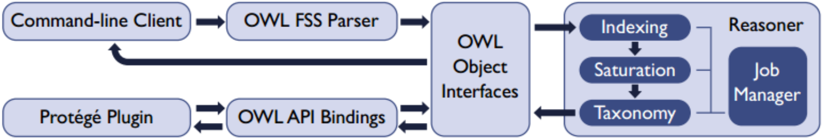


Example: ELK

ELK – state-of-the-art consequence-based reasoner for $\mathcal{EL}$-family

Example: ELK

ELK – state-of-the-art consequence-based reasoner for $\mathcal{EL}$-family



Distinctive features:

- Pipelining: loading | indexing, classificiation | taxonomy
- Concurrency: all concepts are classified in parallel

Stuff Seen Today

Stuff Seen Today

Brief history of Description Logics

- Concept languages originating from semantic networks
- Have formal first-order semantics

Stuff Seen Today

Brief history of Description Logics

- Concept languages originating from semantic networks
- Have formal first-order semantics

$\mathcal{ALC}$ – a basic propositionally complete DL

- Syntax and semantics (concept constructors, model theory)
- TBox and ABox a.k.a. schema and data
- Reasoning problems
  - Concept satisfiability, entailment, ontology consistency
  - Inter-reducibility

Stuff Seen Today

Brief history of Description Logics

- Concept languages originating from semantic networks
- Have formal first-order semantics

$\mathcal{ALC}$ – a basic propositionally complete DL

- Syntax and semantics (concept constructors, model theory)
- TBox and ABox a.k.a. schema and data
- Reasoning problems
  - Concept satisfiability, entailment, ontology consistency
  - Inter-reducibility

What's inside a modern DL reasoner

Stuff Seen Today

Brief history of Description Logics

- Concept languages originating from semantic networks
- Have formal first-order semantics

$\mathcal{ALC}$ – a basic propositionally complete DL

- Syntax and semantics (concept constructors, model theory)
- TBox and ABox a.k.a. schema and data
- Reasoning problems
  - Concept satisfiability, entailment, ontology consistency
  - Inter-reducibility

What's inside a modern DL reasoner

Tomorrow: how reasoning is actually done (tableau algorithms)