Description Logics – Basics

Petr Křemen

petr.kremen@fel.cvut.cz

November 14, 2018

Outline

Towards Description Logics

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain
- background for many AI techniques, e.g.:

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain
- background for many AI techniques, e.g.:
  - knowledge management – search engines, data integration

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain
- background for many AI techniques, e.g.:
  - knowledge management – search engines, data integration
  - multiagent systems – communication between agents

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain
- background for many AI techniques, e.g.:
  - knowledge management – search engines, data integration
  - multiagent systems – communication between agents
  - machine learning – language bias

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain
- background for many AI techniques, e.g.:
  - knowledge management – search engines, data integration
  - multiagent systems – communication between agents
  - machine learning – language bias
- involves many graphical/textual languages ranging from informal to formal ones, e.g. *relational algebra*, *Prolog*, *RDFS*, *OWL*, *topic maps*, *thesauri*, *conceptual graphs*

Formal Ontologies

- deal with proper representation of conceptual knowledge in a domain
- background for many AI techniques, e.g.:
  - knowledge management – search engines, data integration
  - multiagent systems – communication between agents
  - machine learning – language bias
- involves many graphical/textual languages ranging from informal to formal ones, e.g. *relational algebra*, *Prolog*, *RDFS*, *OWL*, *topic maps*, *thesauri*, *conceptual graphs*
- Most of them are based on some **logical calculus**.

Logics for Ontologies

- propositional logic

Logics for Ontologies

- propositional logic

Example

"John is clever." $\Rightarrow$ $\neg$"John fails at exam."

Logics for Ontologies

- propositional logic

Example

"John is clever." $\Rightarrow \neg$"John fails at exam."

- first order predicate logic

Logics for Ontologies

- propositional logic

Example

"John is clever." $\Rightarrow \neg$"John fails at exam."

- first order predicate logic

Example

$(\forall x)(Clever(x) \Rightarrow \neg((\exists y)(Exam(y) \wedge Fails(x, y))))$.

Logics for Ontologies

- propositional logic

Example

"John is clever." $\Rightarrow \neg$"John fails at exam."

- first order predicate logic

Example

$(\forall x)(\mathit{Clever}(x) \Rightarrow \neg((\exists y)(\mathit{Exam}(y) \wedge \mathit{Fails}(x, y))))$.

- (propositional) modal logic

Logics for Ontologies

- propositional logic

Example

"John is clever." $\Rightarrow \neg$"John fails at exam."

- first order predicate logic

Example

$(\forall x)(Clever(x) \Rightarrow \neg((\exists y)(Exam(y) \wedge Fails(x, y))))$.

- (propositional) modal logic

Example

$\Box((\forall x)(Clever(x) \Rightarrow \Diamond\neg((\exists y)(Exam(y) \wedge Fails(x, y)))))$.

Logics for Ontologies

- propositional logic

Example

"John is clever." $\Rightarrow$ $\neg$"John fails at exam."

- first order predicate logic

Example

$(\forall x)(\mathit{Clever}(x) \Rightarrow \neg((\exists y)(\mathit{Exam}(y) \wedge \mathit{Fails}(x,y)))).$

- (propositional) modal logic

Example

$\Box((\forall x)(\mathit{Clever}(x) \Rightarrow \Diamond\neg((\exists y)(\mathit{Exam}(y) \wedge \mathit{Fails}(x,y))))).$

- ... what is the meaning of these formulas ?

Logics for Ontologies (2)

Logics are defined by their

- Syntax – to *represent* concepts (*defining symbols*)

Logics trade-off

A logical calculus is always a trade-off between *expressiveness* and *tractability of reasoning*.

Logics for Ontologies (2)

Logics are defined by their

- Syntax – to *represent* concepts (*defining symbols*)
- Semantics – to capture meaning of the syntactic constructs (*defining concepts*)

Logics trade-off

A logical calculus is always a trade-off between *expressiveness* and *tractability of reasoning*.

Logics for Ontologies (2)

Logics are defined by their

- Syntax – to *represent* concepts (*defining symbols*)
- Semantics – to capture meaning of the syntactic constructs (*defining concepts*)
- Proof Theory – to enforce the semantics

Logics trade-off

A logical calculus is always a trade-off between *expressiveness* and *tractability of reasoning*.

Propositional Logic

Example

How to check satisfiability of the formula $A \vee (\neg(B \wedge A) \vee B \wedge C)$?

syntax – atomic formulas and $\neg$, $\wedge$, $\vee$, $\Rightarrow$

Propositional Logic

Example

How to check satisfiability of the formula $A \vee (\neg(B \wedge A) \vee B \wedge C)$?

syntax – atomic formulas and $\neg$, $\wedge$, $\vee$, $\Rightarrow$

semantics ($\models$) – an interpretation assigns true/false to each formula.

Propositional Logic

Example

How to check satisfiability of the formula $A \vee (\neg(B \wedge A) \vee B \wedge C)$?

syntax – atomic formulas and $\neg$, $\wedge$, $\vee$, $\Rightarrow$

semantics ($\models$) – an interpretation assigns true/false to each formula.

proof theory ($\vdash$) – resolution, tableau

Propositional Logic

Example

How to check satisfiability of the formula $A \vee (\neg(B \wedge A) \vee B \wedge C)$?

syntax – atomic formulas and $\neg$, $\wedge$, $\vee$, $\Rightarrow$

semantics ($\models$) – an interpretation assigns true/false to each formula.

proof theory ($\vdash$) – resolution, tableau

complexity – NP-Complete (Cook theorem)

First Order Predicate Logic

Example

What is the meaning of this sentence ?

$$(\forall x_1)((Student(x_1) \wedge (\exists x_2)(GraduateCourse(x_2) \wedge isEnrolledTo(x_1, x_2))) \Rightarrow (\forall x_3)(isEnrolledTo(x_1, x_3) \Rightarrow GraduateCourse(x_3)))$$

$$Student \sqcap \exists isEnrolledTo.GraduateCourse \sqsubseteq \forall isEnrolledTo.GraduateCourse$$

First Order Predicate Logic – quick informal review

syntax – constructs involve

First Order Predicate Logic – quick informal review

syntax – constructs involve

term (variable x, constant symbol $JOHN$, function symbol applied to terms $fatherOf(JOHN)$)

First Order Predicate Logic – quick informal review

syntax – constructs involve

- term (variable x, constant symbol $JOHN$, function symbol applied to terms $fatherOf(JOHN)$)
- axiom/formula (predicate symbols applied to terms $hasFather(x, JOHN)$, possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)

First Order Predicate Logic – quick informal review

syntax – constructs involve

- term (variable x, constant symbol $JOHN$, function symbol applied to terms $fatherOf(JOHN)$)
- axiom/formula (predicate symbols applied to terms $hasFather(x, JOHN)$, possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)
- universally closed formula formula without free variable $((\forall x)(\exists y)hasFather(x, y) \wedge Person(y))$

First Order Predicate Logic – quick informal review

syntax – constructs involve

term (variable x, constant symbol $JOHN$, function symbol applied to terms $fatherOf(JOHN)$)

axiom/formula (predicate symbols applied to terms $hasFather(x, JOHN)$, possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)

universally closed formula formula without free variable $((\forall x)(\exists y) hasFather(x, y) \wedge Person(y))$

semantics – an interpretation (with valuation) assigns:

First Order Predicate Logic – quick informal review

syntax – constructs involve

term (variable x, constant symbol *JOHN*, function symbol applied to terms $fatherOf(JOHN)$)

axiom/formula (predicate symbols applied to terms $hasFather(x, JOHN)$, possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)

universally closed formula formula without free variable $((\forall x)(\exists y)hasFather(x, y) \wedge Person(y))$

semantics – an interpretation (with valuation) assigns:

domain element to each term

First Order Predicate Logic – quick informal review

syntax – constructs involve

- **term** (variable x, constant symbol *JOHN*, function symbol applied to terms $fatherOf(JOHN)$)
- **axiom/formula** (predicate symbols applied to terms $hasFather(x, JOHN)$, possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)
- **universally closed formula** formula without free variable $((\forall x)(\exists y)hasFather(x, y) \wedge Person(y))$

semantics – an interpretation (with valuation) assigns:

- **domain element** to each term
- **true/false** to each closed formula

First Order Predicate Logic – quick informal review

syntax – constructs involve

term (variable x, constant symbol *JOHN*, function symbol applied to terms $fatherOf(JOHN)$)

axiom/formula (predicate symbols applied to terms $hasFather(x, JOHN)$, possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)

universally closed formula formula without free variable $((\forall x)(\exists y)hasFather(x, y) \wedge Person(y))$

semantics – an interpretation (with valuation) assigns:

domain element to each term

true/false to each closed formula

proof theory – resolution; *Deduction Theorem*, *Soundness Theorem*, *Completeness Theorem*

First Order Predicate Logic – quick informal review

syntax – constructs involve

- term (variable x, constant symbol *JOHN*, function symbol applied to terms *fatherOf*(*JOHN*))
- axiom/formula (predicate symbols applied to terms *hasFather*(x, *JOHN*), possibly glued together with $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$,$\exists$)
- universally closed formula formula without free variable $((\forall x)(\exists y) hasFather(x, y) \wedge Person(y))$

semantics – an interpretation (with valuation) assigns:

- domain element to each term
- true/false to each closed formula

proof theory – resolution; *Deduction Theorem*, *Soundness Theorem*, *Completeness Theorem*

complexity – undecidable (Goedel)

Open World Assumption

OWA

FOPL accepts Open World Assumption, i.e. whatever is not known is not necessarily false.

As a result, FOPL is *monotonic*, i.e.

monotonicity

No conclusion can be invalidated by adding extra knowledge.

This is in contrary to relational databases, or Prolog that accept Closed World Assumption.

Towards Description Logics

Languages sketched so far aren't enough ?

- Why not First Order Predicate Logic ?

Languages sketched so far aren't enough ?

- Why not First Order Predicate Logic ?
  - ☹ FOPL is undecidable – many logical consequences cannot be verified in finite time.

Languages sketched so far aren't enough ?

- Why not First Order Predicate Logic ?
  - ☹ FOPL is undecidable – many logical consequences cannot be verified in finite time.
  - We often do not need full expressiveness of FOL.

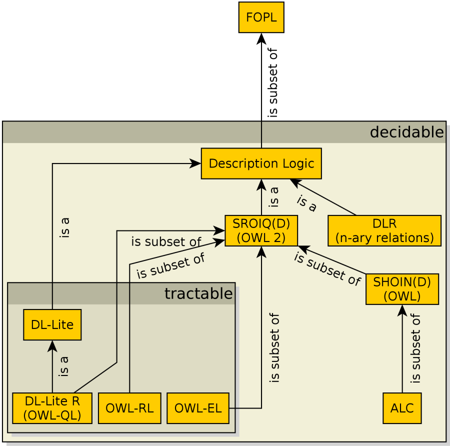
Languages sketched so far aren't enough ?

- Why not First Order Predicate Logic ?
  - ☹ FOPL is undecidable – many logical consequences cannot be verified in finite time.
  - We often do not need full expressiveness of FOL.
- Well, we have Prolog – wide-spread and optimized implementation of FOPL, right ?

Languages sketched so far aren't enough ?

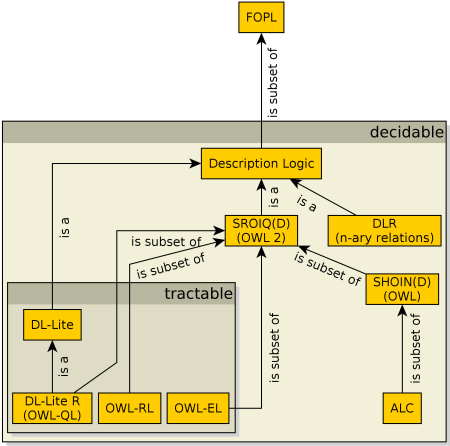
- Why not First Order Predicate Logic ?
  - ☹ FOPL is undecidable – many logical consequences cannot be verified in finite time.
  - We often do not need full expressiveness of FOL.
- Well, we have Prolog – wide-spread and optimized implementation of FOPL, right ?
  - ☹ Prolog is not an implementation of FOPL – OWA vs. CWA, negation as failure, problems in expressing disjunctive knowledge, etc.

What are Description Logics ?



What are Description Logics ?

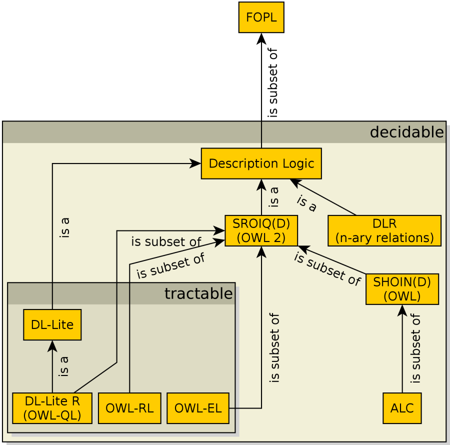
Description logics (DLs) are (almost exclusively) decidable subsets of FOPL aimed at modeling *terminological incomplete knowledge*.



What are Description Logics ?

Description logics (DLs) are (almost exclusively) decidable subsets of FOPL aimed at modeling *terminological incomplete knowledge*.

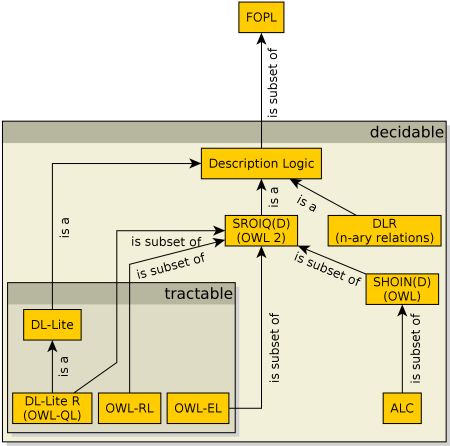
- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.



What are Description Logics ?

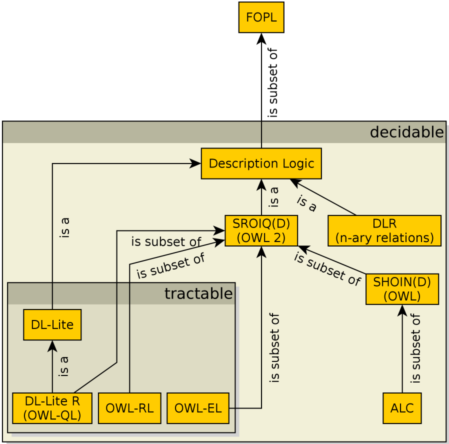
Description logics (DLs) are (almost exclusively) decidable subsets of FOPL aimed at modeling *terminological incomplete knowledge*.

- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.
- 90's $\mathcal{ALC}$



What are Description Logics ?

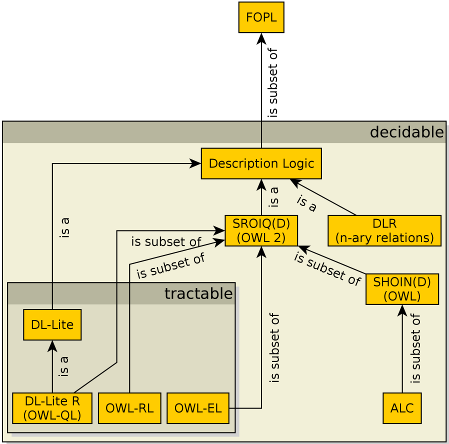
Description logics (DLs) are (almost exclusively) decidable subsets of FOPL aimed at modeling *terminological incomplete knowledge*.



- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.
- 90's $\mathcal{ALC}$
- 2004 $\mathcal{SHOIN}(\mathcal{D})$ – OWL

What are Description Logics ?

Description logics (DLs) are (almost exclusively) decidable subsets of FOPL aimed at modeling *terminological incomplete knowledge*.



- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.
- 90's $\mathcal{ALC}$
- 2004 $\mathcal{SHOIN}(\mathcal{D})$ – OWL
- 2009 $\mathcal{SROIQ}(\mathcal{D})$ – OWL 2

$\mathcal{ALC}$ Language

Concepts and Roles

- Basic building blocks of DLs are :

Concepts and Roles

- Basic building blocks of DLs are :

  (atomic) concepts - representing (named) *unary predicates* / classes, e.g. $Parent$, or $Person \sqcap \exists hasChild \cdot Person$.

Concepts and Roles

- Basic building blocks of DLs are :

  (atomic) concepts - representing (named) *unary predicates* / classes, e.g. *Parent*, or $Person \sqcap \exists hasChild \cdot Person$.

  (atomic) roles - represent (named) *binary predicates* / relations, e.g. *hasChild*

Concepts and Roles

- Basic building blocks of DLs are :

  (atomic) concepts - representing (named) *unary predicates* / classes, e.g. *Parent*, or $Person \sqcap \exists hasChild \cdot Person$.

  (atomic) roles - represent (named) *binary predicates* / relations, e.g. *hasChild*

  individuals - represent ground terms / individuals, e.g. *JOHN*

Concepts and Roles

- Basic building blocks of DLs are :

  (atomic) concepts - representing (named) *unary predicates* / classes, e.g. *Parent*, or $Person \sqcap \exists hasChild \cdot Person$.

  (atomic) roles - represent (named) *binary predicates* / relations, e.g. *hasChild*

  individuals - represent ground terms / individuals, e.g. *JOHN*

- Theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (in OWL refered as Ontology) consists of a

Concepts and Roles

- Basic building blocks of DLs are :

  (atomic) concepts - representing (named) *unary predicates* / classes, e.g. *Parent*, or $Person \sqcap \exists hasChild \cdot Person$.

  (atomic) roles - represent (named) *binary predicates* / relations, e.g. *hasChild*

  individuals - represent ground terms / individuals, e.g. *JOHN*

- Theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (in OWL refered as Ontology) consists of a

  TBOX $\mathcal{T}$ - representing axioms generally valid in the domain, e.g. $\mathcal{T} = \{Man \sqsubseteq Person\}$

Concepts and Roles

- Basic building blocks of DLs are :

  (atomic) concepts - representing (named) *unary predicates* / classes, e.g. *Parent*, or $Person \sqcap \exists hasChild \cdot Person$.

  (atomic) roles - represent (named) *binary predicates* / relations, e.g. *hasChild*

  individuals - represent ground terms / individuals, e.g. *JOHN*

- Theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (in OWL refered as Ontology) consists of a

  TBOX $\mathcal{T}$ - representing axioms generally valid in the domain, e.g. $\mathcal{T} = \{Man \sqsubseteq Person\}$

  ABOX $\mathcal{A}$ - representing a particular relational structure (data), e.g. $\mathcal{A} = \{Man(JOHN), loves(JOHN, MARY)\}$

Concepts and Roles

- Basic building blocks of DLs are :
  - (atomic) concepts - representing (named) *unary predicates* / classes, e.g. *Parent*, or $Person \sqcap \exists hasChild \cdot Person$.
  - (atomic) roles - represent (named) *binary predicates* / relations, e.g. *hasChild*
  - individuals - represent ground terms / individuals, e.g. *JOHN*
- Theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (in OWL refered as Ontology) consists of a
  - TBOX $\mathcal{T}$ - representing axioms generally valid in the domain, e.g. $\mathcal{T} = \{Man \sqsubseteq Person\}$
  - ABOX $\mathcal{A}$ - representing a particular relational structure (data), e.g. $\mathcal{A} = \{Man(JOHN), loves(JOHN, MARY)\}$
- DLs differ in their expressive power (concept/role constructors, axiom types).

Semantics, Interpretation

- as $\mathcal{ALC}$ is a subset of FOPL, let's define semantics analogously (and restrict interpretation function where applicable):

Semantics, Interpretation

- as $\mathcal{ALC}$ is a subset of FOPL, let's define semantics analogously (and restrict interpretation function where applicable):
- **Interpretation** is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is an interpretation domain and $\cdot^{\mathcal{I}}$ is an interpretation function.

Semantics, Interpretation

- as $\mathcal{ALC}$ is a subset of FOPL, let's define semantics analogously (and restrict interpretation function where applicable):
- **Interpretation** is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is an interpretation domain and $\cdot^{\mathcal{I}}$ is an interpretation function.
- Having *atomic* concept A, *atomic* role R and individual a, then

$$
\begin{aligned}
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
R^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
a^{\mathcal{I}} &\in \Delta^{\mathcal{I}}
\end{aligned}
$$

$\mathcal{ALC}$ (= attributive language with complements)

Having concepts C, D, atomic concept A and atomic role R, then for interpretation $\mathcal{I}$:

concept	*concept*$^{\mathcal{I}}$	*description*
$\top$	$\Delta^{\mathcal{I}}$	(universal concept)
$\bot$	$\emptyset$	(unsatisfiable concept)
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	(negation)
$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(intersection)
$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	(union)
$\forall R \cdot C$	$\{a \mid \forall b((a,b) \in R^{\mathcal{I}} \implies b \in C^{\mathcal{I}})\}$	(universal restriction)
$\exists R \cdot C$	$\{a \mid \exists b((a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}})\}$	(existential restriction)

[1] two different individuals denote two different domain elements

$\mathcal{ALC}$ (= attributive language with complements)

Having concepts C, D, atomic concept A and atomic role R, then for interpretation $\mathcal{I}$:

concept	*concept*$^{\mathcal{I}}$	*description*
$\top$	$\Delta^{\mathcal{I}}$	(universal concept)
$\bot$	$\emptyset$	(unsatisfiable concept)
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	(negation)
$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(intersection)
$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	(union)
$\forall R \cdot C$	$\{a \mid \forall b((a,b) \in R^{\mathcal{I}} \implies b \in C^{\mathcal{I}})\}$	(universal restriction)
$\exists R \cdot C$	$\{a \mid \exists b((a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}})\}$	(existential restriction)

TBOX

axiom	$\mathcal{I} \models$ axiom iff	*description*
$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$	(inclusion)
$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$	(equivalence)

[1] two different individuals denote two different domain elements

$\mathcal{ALC}$ (= attributive language with complements)

Having concepts C, D, atomic concept A and atomic role R, then for interpretation $\mathcal{I}$:

concept	*concept*$^{\mathcal{I}}$	*description*
$\top$	$\Delta^{\mathcal{I}}$	(universal concept)
$\bot$	$\emptyset$	(unsatisfiable concept)
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	(negation)
$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(intersection)
$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	(union)
$\forall R \cdot C$	$\{a \mid \forall b((a,b) \in R^{\mathcal{I}} \implies b \in C^{\mathcal{I}})\}$	(universal restriction)
$\exists R \cdot C$	$\{a \mid \exists b((a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}})\}$	(existential restriction)

TBOX

axiom	$\mathcal{I} \models$ axiom iff	*description*
$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$	(inclusion)
$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$	(equivalence)

ABOX (UNA = unique name assumption[1])

axiom	$\mathcal{I} \models$ axiom iff	*description*
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	(concept assertion)
$R(a_1, a_2)$	$(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}$	(role assertion)

[1] two different individuals denote two different domain elements

Logical Consequence

For an arbitrary set S of axioms (resp. theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $S = \mathcal{T} \cup \mathcal{A}$) :

Logical Consequence

For an arbitrary set S of axioms (resp. theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $S = \mathcal{T} \cup \mathcal{A}$) :

Model

$\mathcal{I} \models S$ if $\mathcal{I} \models \alpha$ for all $\alpha \in S$ ($\mathcal{I}$ is a model of S, resp. $\mathcal{K}$)

Logical Consequence

For an arbitrary set S of axioms (resp. theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $S = \mathcal{T} \cup \mathcal{A}$) :

Model

$\mathcal{I} \models S$ if $\mathcal{I} \models \alpha$ for all $\alpha \in S$ ($\mathcal{I}$ is a model of S, resp. $\mathcal{K}$)

Logical Consequence

For an arbitrary set S of axioms (resp. theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $S = \mathcal{T} \cup \mathcal{A}$) :

Model

$\mathcal{I} \models S$ if $\mathcal{I} \models \alpha$ for all $\alpha \in S$ ($\mathcal{I}$ is a model of S, resp. $\mathcal{K}$)

Logical Consequence

$S \models \beta$ if $\mathcal{I} \models \beta$ whenever $\mathcal{I} \models S$ (β is a logical consequence of S, resp. $\mathcal{K}$)

Logical Consequence

For an arbitrary set S of axioms (resp. theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $S = \mathcal{T} \cup \mathcal{A}$) :

Model

$\mathcal{I} \models S$ if $\mathcal{I} \models \alpha$ for all $\alpha \in S$ ($\mathcal{I}$ is a model of S, resp. $\mathcal{K}$)

Logical Consequence

$S \models \beta$ if $\mathcal{I} \models \beta$ whenever $\mathcal{I} \models S$ (β is a logical consequence of S, resp. $\mathcal{K}$)

- S is consistent, if S has at least one model

$\mathcal{ALC}$ – Example

Example

Consider an information system for genealogical data. Information integration from various sources is crucial – databases, information systems with *different data models*. As an integration layer, let's use a description logic theory. Let's have atomic concepts *Person*, *Man*, *GrandParent* and atomic role *hasChild*.

- Set of persons that have just men as their descendants, if any ? (specify a *concept*)

$\mathcal{ALC}$ – Example

Example

Consider an information system for genealogical data. Information integration from various sources is crucial – databases, information systems with *different data models*. As an integration layer, let's use a description logic theory. Let's have atomic concepts *Person*, *Man*, *GrandParent* and atomic role *hasChild*.

- Set of persons that have just men as their descendants, if any ? (specify a *concept*)
  - $Person \sqcap \forall hasChild \cdot Man$

$\mathcal{ALC}$ – Example

Example

Consider an information system for genealogical data. Information integration from various sources is crucial – databases, information systems with *different data models*. As an integration layer, let's use a description logic theory. Let's have atomic concepts *Person*, *Man*, *GrandParent* and atomic role *hasChild*.

- Set of persons that have just men as their descendants, if any ? (specify a *concept*)
  - $Person \sqcap \forall hasChild \cdot Man$
- How to define concept *GrandParent* ? (specify an *axiom*)

$\mathcal{ALC}$ – Example

Example

Consider an information system for genealogical data. Information integration from various sources is crucial – databases, information systems with *different data models*. As an integration layer, let's use a description logic theory. Let's have atomic concepts *Person*, *Man*, *GrandParent* and atomic role *hasChild*.

- Set of persons that have just men as their descendants, if any ? (specify a *concept*)
  - $Person \sqcap \forall hasChild \cdot Man$
- How to define concept *GrandParent* ? (specify an *axiom*)
  - $GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top$

$\mathcal{ALC}$ – Example

Example

Consider an information system for genealogical data. Information integration from various sources is crucial – databases, information systems with *different data models*. As an integration layer, let's use a description logic theory. Let's have atomic concepts *Person*, *Man*, *GrandParent* and atomic role *hasChild*.

- Set of persons that have just men as their descendants, if any ? (specify a *concept*)
  - $Person \sqcap \forall hasChild \cdot Man$
- How to define concept *GrandParent* ? (specify an *axiom*)
  - $GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top$
- How does the previous axiom look like in FOPL ?

$$\forall x \, (GrandParent(x) \equiv (Person(x) \wedge \exists y \, (hasChild(x, y) \\ \wedge \exists z \, (hasChild(y, z)))))$$

$\mathcal{ALC}$ Example – $\mathcal{T}$

Example

$$
\begin{aligned}
Woman &\equiv Person \sqcap Female \\
Man &\equiv Person \sqcap \neg Woman \\
Mother &\equiv Woman \sqcap \exists hasChild \cdot Person \\
Father &\equiv Man \sqcap \exists hasChild \cdot Person \\
Parent &\equiv Father \sqcup Mother \\
Grandmother &\equiv Mother \sqcap \exists hasChild \cdot Parent \\
MotherWithoutDaughter &\equiv Mother \sqcap \forall hasChild \cdot \neg Woman \\
Wife &\equiv Woman \sqcap \exists hasHusband \cdot Man
\end{aligned}
$$

Interpretation – Example

Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.

Interpretation – Example

Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.
- a model of $\mathcal{K}_1$ can be interpretation $\mathcal{I}_1$:

Interpretation – Example

Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.
- a model of $\mathcal{K}_1$ can be interpretation $\mathcal{I}_1$:
  - $\Delta^{\mathcal{I}_1} = Man^{\mathcal{I}_1} = Person^{\mathcal{I}_1} = \{John, Phillipe, Martin\}$

Interpretation – Example

Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.
- a model of $\mathcal{K}_1$ can be interpretation $\mathcal{I}_1$:
    - $\Delta^{\mathcal{I}_1} = Man^{\mathcal{I}_1} = Person^{\mathcal{I}_1} = \{John, Phillipe, Martin\}$
    - $hasChild^{\mathcal{I}_1} = \{(John, Phillipe), (Phillipe, Martin)\}$

Interpretation – Example

Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.
- a model of $\mathcal{K}_1$ can be interpretation $\mathcal{I}_1$:
  - $\Delta^{\mathcal{I}_1} = Man^{\mathcal{I}_1} = Person^{\mathcal{I}_1} = \{John, Phillipe, Martin\}$
  - $hasChild^{\mathcal{I}_1} = \{(John, Phillipe), (Phillipe, Martin)\}$
  - $GrandParent^{\mathcal{I}_1} = \{John\}$

Interpretation – Example

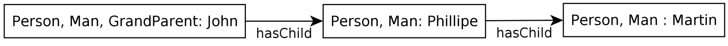
Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.
- a model of $\mathcal{K}_1$ can be interpretation $\mathcal{I}_1$:
  - $\Delta^{\mathcal{I}_1} = Man^{\mathcal{I}_1} = Person^{\mathcal{I}_1} = \{John, Phillipe, Martin\}$
  - $hasChild^{\mathcal{I}_1} = \{(John, Phillipe), (Phillipe, Martin)\}$
  - $GrandParent^{\mathcal{I}_1} = \{John\}$
  - $JOHN^{\mathcal{I}_1} = \{John\}$

Interpretation – Example

Example

- Consider a theory $\mathcal{K}_1 = (\{GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top\}, \{GrandParent(JOHN)\})$. Find some model.
- a model of $\mathcal{K}_1$ can be interpretation $\mathcal{I}_1$:
  - $\Delta^{\mathcal{I}_1} = Man^{\mathcal{I}_1} = Person^{\mathcal{I}_1} = \{John, Phillipe, Martin\}$
  - $hasChild^{\mathcal{I}_1} = \{(John, Phillipe), (Phillipe, Martin)\}$
  - $GrandParent^{\mathcal{I}_1} = \{John\}$
  - $JOHN^{\mathcal{I}_1} = \{John\}$
- this model is finite and has the form of a tree with the root in the node *John* :



Shape of DL Models

The last example revealed several important properties of DL models:

Shape of DL Models

The last example revealed several important properties of DL models:

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property (TMP)

Every consistent $\mathcal{K} = (\{\}, \{C(I)\})$ has a model in the shape of a *rooted tree*.

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property (TMP)

Every consistent $\mathcal{K} = (\{\}, \{C(I)\})$ has a model in the shape of a *rooted tree*.

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property (TMP)

Every consistent $\mathcal{K} = (\{\}, \{C(I)\})$ has a model in the shape of a *rooted tree*.

Finite model property (FMP)

Every consistent $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a *finite model*.

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property (TMP)

Every consistent $\mathcal{K} = (\{\}, \{C(I)\})$ has a model in the shape of a *rooted tree*.

Finite model property (FMP)

Every consistent $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a *finite model*.

Both properties represent important characteristics of $\mathcal{ALC}$ that significantly speed-up reasoning.

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property (TMP)

Every consistent $\mathcal{K} = (\{\}, \{C(I)\})$ has a model in the shape of a *rooted tree*.

Finite model property (FMP)

Every consistent $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a *finite model*.

- Both properties represent important characteristics of $\mathcal{ALC}$ that significantly speed-up reasoning.
- In particular (generalized) TMP is a characteristics that is shared by most DLs and significantly reduces their computational complexity.

Example – CWA × OWA

Example

ABOX

$hasChild(JOCASTA, OEDIPUS)$
$hasChild(OEDIPUS, POLYNEIKES)$
$Patricide(OEDIPUS)$
$hasChild(JOCASTA, POLYNEIKES)$
$hasChild(POLYNEIKES, THERSANDROS)$
$\neg Patricide(THERSANDROS)$

Example – CWA × OWA

Example

ABOX

$hasChild(JOCASTA, OEDIPUS)$
$hasChild(OEDIPUS, POLYNEIKES)$
$Patricide(OEDIPUS)$
$hasChild(JOCASTA, POLYNEIKES)$
$hasChild(POLYNEIKES, THERSANDROS)$
$\neg Patricide(THERSANDROS)$

Edges represent role assertions of *hasChild*; red/green colors distinguish concepts instances – *Patricide* a $\neg$*Patricide*

Example – CWA × OWA

Example

ABOX

hasChild(*JOCASTA*, *OEDIPUS*)	*hasChild*(*JOCASTA*, *POLYNEIKES*)
hasChild(*OEDIPUS*, *POLYNEIKES*)	*hasChild*(*POLYNEIKES*, *THERSANDROS*)
Patricide(*OEDIPUS*)	¬*Patricide*(*THERSANDROS*)

Edges represent role assertions of *hasChild*; red/green colors distinguish concepts instances – *Patricide* a ¬*Patricide*

Q1 $(\exists hasChild \cdot (Patricide \sqcap \exists hasChild \cdot \neg Patricide))(JOCASTA)$,

Example – CWA × OWA

Example

ABOX

$hasChild(JOCASTA, OEDIPUS)$ $\quad$ $hasChild(JOCASTA, POLYNEIKES)$
$hasChild(OEDIPUS, POLYNEIKES)$ $\quad$ $hasChild(POLYNEIKES, THERSANDROS)$
$Patricide(OEDIPUS)$ $\quad$ $\neg Patricide(THERSANDROS)$

Edges represent role assertions of *hasChild*; red/green colors distinguish concepts instances – *Patricide* a $\neg$*Patricide*

Q1 $(\exists hasChild \cdot (Patricide \sqcap \exists hasChild \cdot \neg Patricide))(JOCASTA)$,

$$JOCASTA \longrightarrow \bullet \longrightarrow \bullet$$

Q2 Find individuals x such that $\mathcal{K} \models C(x)$, where C is

$$\neg Patricide \sqcap \exists hasChild^{-} \cdot (Patricide \sqcap \exists hasChild^{-}) \cdot \{JOCASTA\}$$

What is the difference, when considering CWA ?

$$JOCASTA \longrightarrow \bullet \longrightarrow x$$

From $\mathcal{ALC}$ to OWL(2)-DL

Extending . . . $\mathcal{ALC}$ ...

- We have introduced $\mathcal{ALC}$, together with a decision procedure. Its expressiveness is higher than propositional calculus, still it is insufficient for many practical applications.

Extending . . . $\mathcal{ALC}$ ...

- We have introduced $\mathcal{ALC}$, together with a decision procedure. Its expressiveness is higher than propositional calculus, still it is insufficient for many practical applications.
- Let's take a look, how to extend $\mathcal{ALC}$ while preserving decidability.

Extending . . . $\mathcal{ALC}$ ... (2)

$\mathcal{N}$ (Number restructions) are used for restricting the number of successors in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\, R)$	$\left\{ a \middle\vert\ \left\vert \{ b \mid (a,b) \in R^{\mathcal{I}} \} \right\vert \geq n \right\}$
$(\leq\ n\, R)$	$\left\{ a \middle\vert\ \left\vert \{ b \mid (a,b) \in R^{\mathcal{I}} \} \right\vert \leq n \right\}$
$(=\ n\, R)$	$\left\{ a \middle\vert\ \left\vert \{ b \mid (a,b) \in R^{\mathcal{I}} \} \right\vert = n \right\}$

Example

- Concept $Woman \sqcap (\leq\ 3\ hasChild)$ denotes women who have at most 3 children.

Extending . . . $\mathcal{ALC}$ ... (2)

$\mathcal{N}$ (Number restructions) are used for restricting the number of successors in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\, R)$	$\left\{ a \middle\vert \quad \left\lvert \{b \mid (a,b) \in R^{\mathcal{I}}\} \right\rvert \geq n \right\}$
$(\leq\ n\, R)$	$\left\{ a \middle\vert \quad \left\lvert \{b \mid (a,b) \in R^{\mathcal{I}}\} \right\rvert \leq n \right\}$
$(=\ n\, R)$	$\left\{ a \middle\vert \quad \left\lvert \{b \mid (a,b) \in R^{\mathcal{I}}\} \right\rvert = n \right\}$

Example

- Concept $Woman \sqcap (\leq\ 3\ hasChild)$ denotes women who have at most 3 children.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasWheel)$?

Extending . . . $\mathcal{ALC}$ ... (2)

$\mathcal{N}$ (Number restructions) are used for restricting the number of successors in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\ R)$	$\left\{ a \,\middle|\, \left\lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\rvert \geq n \right\}$
$(\leq\ n\ R)$	$\left\{ a \,\middle|\, \left\lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\rvert \leq n \right\}$
$(=\ n\ R)$	$\left\{ a \,\middle|\, \left\lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\rvert = n \right\}$

Example

- Concept $Woman \sqcap (\leq\ 3\ hasChild)$ denotes women who have at most 3 children.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasWheel)$?
- ... and $Bicycle \equiv (=\ 2\ hasWheel)$?

Extending . . . $\mathcal{ALC}$ ... (3)

$\mathcal{Q}$ (Qualified number restrictions) are used for restricting the number of successors *of the given type* in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\,R\,C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert \geq n\ \right\}$
$(\leq\ n\,R\,C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert \leq n\ \right\}$
$(=\ n\,R\,C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert = n\ \right\}$

Example

- Concept $Woman \sqcap (\geq\ 3\ hasChild\ Man)$ denotes women who have at least 3 sons.

Extending . . . $\mathcal{ALC}$ ... (3)

$\mathcal{Q}$ (Qualified number restrictions) are used for restricting the number of successors *of the given type* in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\,R\,C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert \geq n\ \right\}$
$(\leq\ n\,R\,C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert \leq n\ \right\}$
$(=\ n\,R\,C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert = n\ \right\}$

Example

- Concept $Woman \sqcap (\geq\ 3\ hasChild\ Man)$ denotes women who have at least 3 sons.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasPart\ Wheel)$?

Extending . . . $\mathcal{ALC}$ ... (3)

$\mathcal{Q}$ (Qualified number restrictions) are used for restricting the number of successors *of the given type* in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\,R\,C)$	$\left\{ a \middle\| \ \left\|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\| \geq n \right\}$
$(\leq\ n\,R\,C)$	$\left\{ a \middle\| \ \left\|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\| \leq n \right\}$
$(=\ n\,R\,C)$	$\left\{ a \middle\| \ \left\|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\| = n \right\}$

Example

- Concept $Woman \sqcap (\geq\ 3\ hasChild\ Man)$ denotes women who have at least 3 sons.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasPart\ Wheel)$?
- Which qualified number restrictions can be expressed in $\mathcal{ALC}$?

Extending . . . $\mathcal{ALC}$ ... (4)

$\mathcal{O}$ (Nominals) can be used for naming a concept elements explicitely.

syntax (concept)	semantics
$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Example

- Concept $\{MALE, FEMALE\}$ denotes a gender concept that must be interpreted with at most two elements. Why at most ?

Extending ... $\mathcal{ALC}$ ... (4)

$\mathcal{O}$ (Nominals) can be used for naming a concept elements explicitely.

syntax (concept)	semantics
$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Example

- Concept $\{MALE, FEMALE\}$ denotes a gender concept that must be interpreted with at most two elements. Why at most ?
- $Continent \equiv$ $\{EUROPE, ASIA, AMERICA, AUSTRALIA, AFRICA, ANTARCTICA\}$?

Extending . . . $\mathcal{ALC}$ ... (5)

$\mathcal{I}$ (Inverse roles) are used for defining role inversion.

syntax (role)	semantics
R^-	$(R^{\mathcal{I}})^{-1}$

Example

- Role *hasChild*$^-$ denotes the relationship *hasParent*.

Extending . . . $\mathcal{ALC}$ ... (5)

$\mathcal{I}$ (Inverse roles) are used for defining role inversion.

syntax (role)	semantics
R^-	$(R^{\mathcal{I}})^{-1}$

Example

- Role $hasChild^-$ denotes the relationship $hasParent$.
- What denotes axiom $Person \sqsubseteq (= 2\ hasChild^-)$?

Extending . . . $\mathcal{ALC}$ ... (5)

$\mathcal{I}$ (Inverse roles) are used for defining role inversion.

syntax (role)	semantics
R^-	$(R^\mathcal{I})^{-1}$

Example

- Role $hasChild^-$ denotes the relationship $hasParent$.
- What denotes axiom $Person \sqsubseteq (= 2\ hasChild^-)$?
- What denotes axiom $Person \sqsubseteq \exists hasChild^- \cdot \exists hasChild \cdot \top$?

Extending . . . $\mathcal{ALC}$ ... (6)

$._{trans}$ (Role transitivity axiom) denotes that a role is transitive. Attention – it is not a transitive closure operator.

syntax (axiom)	semantics
$trans(R)$	$R^{\mathcal{I}}$ is transitive

Example

- Role *isPartOf* can be defined as transitive, while role *hasParent* is not. What about roles *hasPart*, $hasPart^{-}$, $hasGrandFather^{-}$?

Extending ... $\mathcal{ALC}$ ... (6)

$._{trans}$ (Role transitivity axiom) denotes that a role is transitive. Attention – it is not a transitive closure operator.

syntax (axiom)	semantics
$trans(R)$	$R^{\mathcal{I}}$ is transitive

Example

- Role *isPartOf* can be defined as transitive, while role *hasParent* is not. What about roles *hasPart*, $hasPart^{-}$, $hasGrandFather^{-}$?
- What is a transitive closure of a relationship ? What is the difference between a transitive closure of $hasDirectBoss^{\mathcal{I}}$ and $hasBoss^{\mathcal{I}}$.

Extending . . . $\mathcal{ALC}$ ...(7)

$\mathcal{H}$ (Role hierarchy) serves for expressing role hierarchies (taxonomies) – similarly to concept hierarchies.

syntax (axiom)	semantics
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

Example

- Role *hasMother* can be defined as a special case of the role *hasParent*.

Extending . . . $\mathcal{ALC}$ ...(7)

$\mathcal{H}$ (Role hierarchy) serves for expressing role hierarchies (taxonomies) – similarly to concept hierarchies.

syntax (axiom)	semantics
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

Example

- Role *hasMother* can be defined as a special case of the role *hasParent*.
- What is the difference between a concept hierarchy $Mother \sqsubseteq Parent$ and role hierarchy $hasMother \sqsubseteq hasParent$.

Extending . . . $\mathcal{ALC}$ ... (8)

$\mathcal{R}$ (role extensions) serve for defining expressive role constructs, like role chains, role disjunctions, etc.

syntax	semantics
$R \circ S \sqsubseteq P$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} \sqsubseteq P^{\mathcal{I}}$
$Dis(R, R)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$\exists R \cdot Self$	$\{a \| (a, a) \in R^{\mathcal{I}}\}$

Example

- How would you define the role *hasUncle* by means of *hasSibling* and *hasParent* ?

Extending . . . $\mathcal{ALC}$ ... (8)

$\mathcal{R}$ (role extensions) serve for defining expressive role constructs, like role chains, role disjunctions, etc.

syntax	semantics
$R \circ S \sqsubseteq P$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} \sqsubseteq P^{\mathcal{I}}$
$Dis(R, R)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$\exists R \cdot Self$	$\{a \mid (a, a) \in R^{\mathcal{I}}\}$

Example

- How would you define the role *hasUncle* by means of *hasSibling* and *hasParent* ?
- how to express that R is transitive, using a role chain ?

Extending ... $\mathcal{ALC}$ ... (8)

$\mathcal{R}$ (role extensions) serve for defining expressive role constructs, like role chains, role disjunctions, etc.

syntax	semantics
$R \circ S \sqsubseteq P$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} \sqsubseteq P^{\mathcal{I}}$
$Dis(R, R)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$\exists R \cdot Self$	$\{a \| (a, a) \in R^{\mathcal{I}}\}$

Example

- How would you define the role *hasUncle* by means of *hasSibling* and *hasParent* ?
- how to express that R is transitive, using a role chain ?
- Whom does the following concept denote $Person \sqcap \exists likes \cdot Self$?

Global restrictions

- *Simple roles* have no (direct or indirect) subroles that are either *transitive* or are defined by means of property chains

$$
\begin{aligned}
hasFather \circ hasBrother &\sqsubseteq hasUncle \\
hasUncle &\sqsubseteq hasRelative \\
hasBiologicalFather &\sqsubseteq hasFather
\end{aligned}
$$

  hasRelative and *hasUncle* are not simple.

- Each concept construct and each axiom from this list contains only *simple roles*:
  - number restrictions – $(\geq\ n\,R)$, $(=\ n\,R)$, $(\leq\ n\,R)$ + their qualified versions
  - $\exists R \cdot Self$
  - specifying functionality/inverse functionality (leads to number restrictions)
  - specifying irreflexivity, asymmetry, and disjoint object properties.

Extending ... $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.

Extending ... $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.

Extending ... $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
    - additional inference rules reflecting the semantics of newly added constructs ($\mathcal{O}, \mathcal{N}, \mathcal{Q}$)

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs ($\mathcal{O}, \mathcal{N}, \mathcal{Q}$)
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. ($\mathcal{H}, \mathcal{R}, \mathcal{I}$)
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Rules and Description Logics

- How to express e.g. that "A cousin is someone whose parent is a sibling of your parent." ?

Rules and Description Logics

- How to express e.g. that "A cousin is someone whose parent is a sibling of your parent." ?
- … we need rules, like

$$hasCousin(?c_1, ?c_2) \leftarrow \quad hasParent(?c_1, ?p_1), hasParent(?c_2, ?p_2), Man(?c_2), hasSibling(?p_1, ?p_2)$$

Rules and Description Logics

- How to express e.g. that "A cousin is someone whose parent is a sibling of your parent." ?
- ... we need rules, like

$$\begin{aligned} hasCousin(?c_1, ?c_2) \leftarrow \quad & hasParent(?c_1, ?p_1), hasParent(?c_2, ?p_2), \\ & Man(?c_2), hasSibling(?p_1, ?p_2) \end{aligned}$$

- in general, each variable can bind **domain elements**; however, such version is *undecidable*.

Rules and Description Logics

- How to express e.g. that "A cousin is someone whose parent is a sibling of your parent." ?
- ... we need rules, like

$$hasCousin(?c_1, ?c_2) \leftarrow hasParent(?c_1, ?p_1), hasParent(?c_2, ?p_2), Man(?c_2), hasSibling(?p_1, ?p_2)$$

- in general, each variable can bind **domain elements**; however, such version is *undecidable*.

DL-safe rules

DL-safe rules are decidable conjunctive rules where each variable **only binds individuals** (not domain elements themselves).

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \qquad (1)$$

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modal propositional logic, where each role represents the accessibility relationa between worlds in Kripke structure, the previous example can be transformed to the modal logic as:

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modal propositional logic, where each role represents the accessibility relationa between worlds in Kripke structure, the previous example can be transformed to the modal logic as:
- $$\Box(Man \implies Person \land \Box_{hasFather} Man) \quad (2)$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions

Data Types ($\mathcal{D}$) allow integrating a data domain (numbers, strings), e.g. $Person \sqcap \exists hasAge \cdot 23$ represents the concept describing "23-years old persons".

References I

[1] * Vladimír Mařík, Olga Štěpánková, and Jiří Lažanský.
Umělá inteligence 6 [in czech], Chapters 2-4.
Academia, 2013.

[2] * Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors.
The Description Logic Handbook, Theory, Implementation and Applications, Chapters 2-4.
Cambridge, 2003.

[3] * Enrico Franconi.
Course on Description Logics.
http://www.inf.unibz.it/ franconi/dl/course/, cit. 22.9.2013.