Introduction, Description Logics

Petr Křemen
petr.kremen@fel.cvut.cz

September 29, 2014

Our plan

Course Information

Course Information

- web page:
  http://cw.felk.cvut.cz/doku.php/courses/ae4m33rzn/start

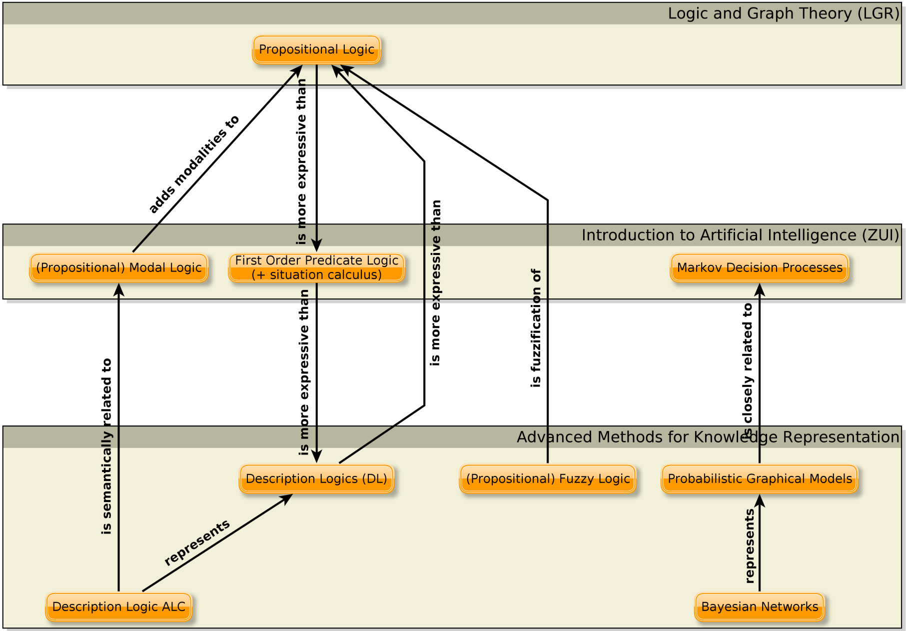
Course Information

- web page: `http://cw.felk.cvut.cz/doku.php/courses/ae4m33rzn/start`
- three basic topics: description logics, fuzzy (description) logic, probabilistic models

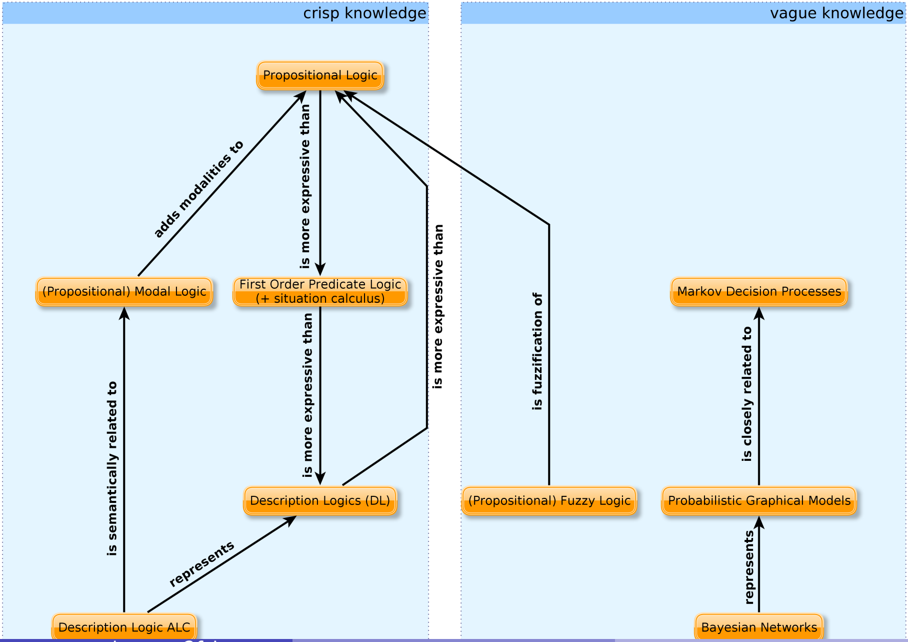
Course Information

- web page: http://cw.felk.cvut.cz/doku.php/courses/ae4m33rzn/start
- three basic topics: description logics, fuzzy (description) logic, probabilistic models
- **Please go through the course web page carefully !!!**

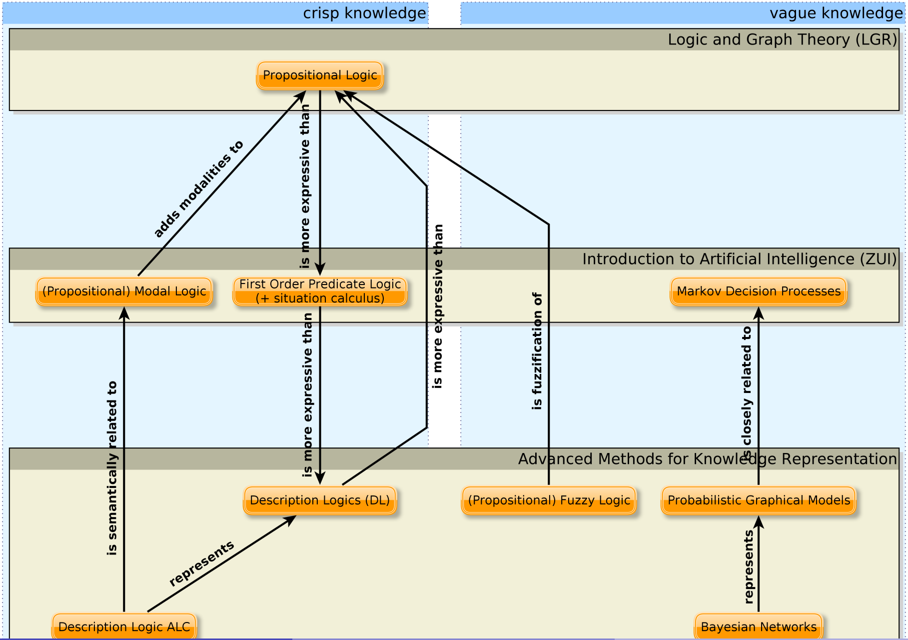
Course Roadmap



Course Roadmap (2)

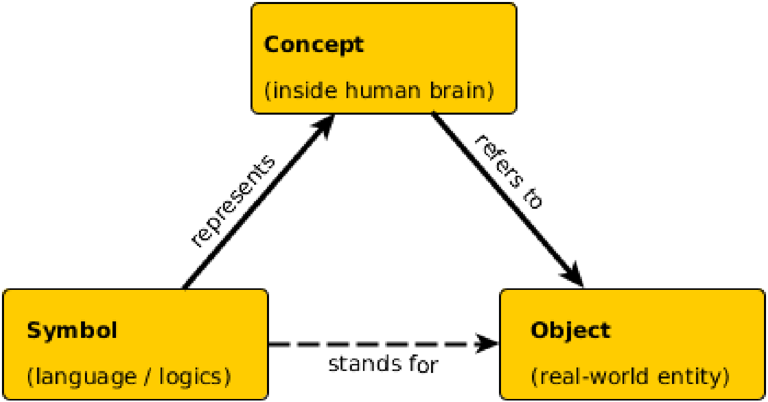


Course Roadmap (3)



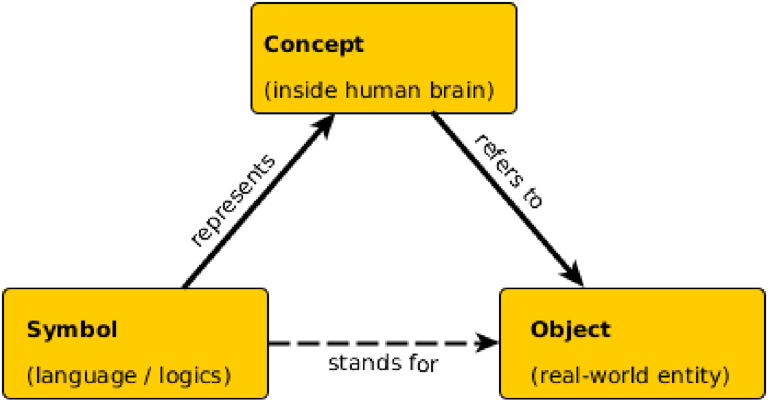
Towards Description Logics

Semiotic Triangle



refers to $\sim$ modeled by *ontologies*; you can learn in AE0M33OSW course

Semiotic Triangle



refers to ∼ modeled by *ontologies*; you can learn in AE0M33OSW course

represents ∼ studied by *formal knowledge representation languages* – **this course**

Knowledge Representation

- deal with proper representation of conceptual knowledge in a domain

Knowledge Representation

- deal with proper representation of conceptual knowledge in a domain
- is used in many AI domains, e.g.:

Knowledge Representation

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

Knowledge Representation

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

Knowledge Representation

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

Knowledge Representation

- deal with proper representation of conceptual knowledge in a domain
- is used in many AI domains, 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*

Knowledge Representation

- deal with proper representation of conceptual knowledge in a domain
- is used in many AI domains, 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 – a Review

Logics for Knowledge Representation

- propositional logic

Logics for Knowledge Representation

- propositional logic

Example

“Everyone is clever.” $\Rightarrow \neg$ “John fails at exam.”

Logics for Knowledge Representation

- propositional logic

Example

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

- first order predicate logic

Logics for Knowledge Representation

- propositional logic

Example

"Everyone 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 Knowledge Representation

- propositional logic

Example

"Everyone 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

Logics for Knowledge Representation

- propositional logic

Example

"Everyone 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 Knowledge Representation

- propositional logic

Example

"Everyone 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

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

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

Logics for KR (2)

Logics are defined by their

- Syntax – to *represent* concepts

Logics trade-off

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

Logics for KR (2)

Logics are defined by their

- Syntax – to *represent* concepts
- Semantics – to capture meaning of the syntactic constructs

Logics trade-off

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

Logics for KR (2)

Logics are defined by their

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

Logics trade-off

A logic 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.

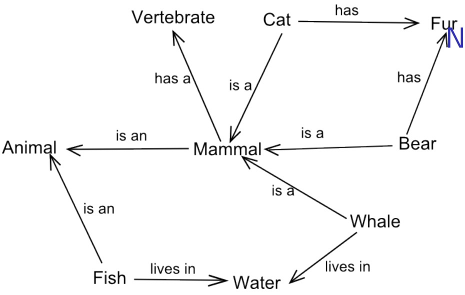
monotonicity

No conclusion can be invalidated by adding extra knowledge.

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

Semantic Networks and Frames

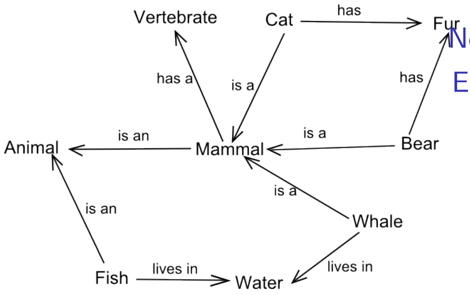
Semantic Networks



(©wikipedia.org)

Nodes = entities (individuals, classes),

Semantic Networks

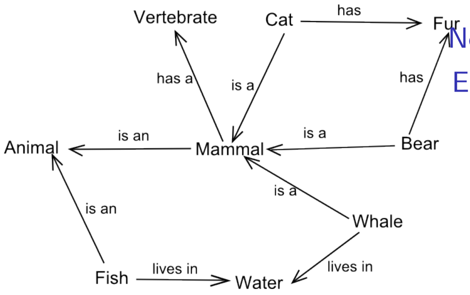


(©wikipedia.org)

Nodes = entities (individuals, classes),

Edges = binary relations

Semantic Networks



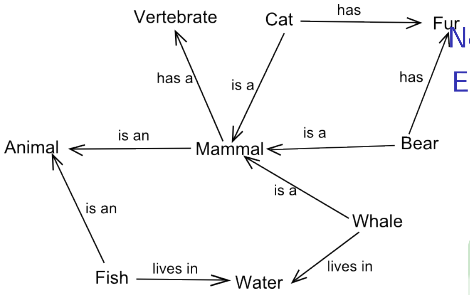
(©wikipedia.org)

Nodes = entities (individuals, classes),

Edges = binary relations

- The only possible inferrence is *inheritance* by means of **is a** relationship.

Semantic Networks



(©wikipedia.org)

Nodes = entities (individuals, classes),

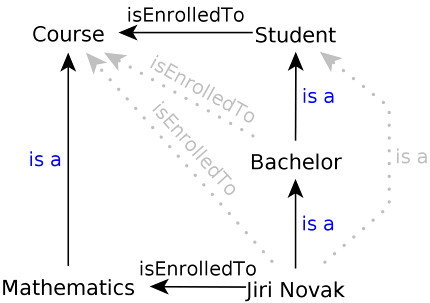
Edges = binary relations

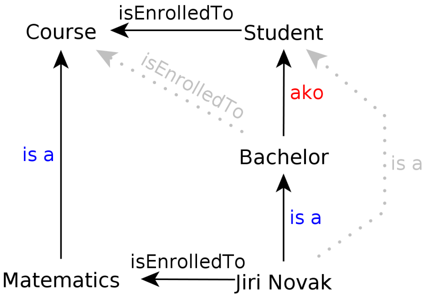
- The only possible inferrence is *inheritance* by means of **is a** relationship.

Example

Each *Cat hasa Vertebrate*, since each *Cat isa Mammal*.

Semantic Networks (2)





However, this does not allow distinguishing individuals (instances) and groups (classes).

To solve this, a new relationship type "is a kind of" **ako** can be introduced and used for inheritance, while **is a** relationships would be restricted to expressing individual-group relationships.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$
$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$
$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$
$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$
$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$
$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$
$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

$$
\begin{aligned}
relation(X,Y) \wedge ako(Z,X) &\Rightarrow relation(Z,Y).\\
isa(X,Y) \wedge ako(Y,Z) &\Rightarrow isa(X,Z).\\
ako(X,Y) \wedge ako(Y,Z) &\Rightarrow ako(X,Z).
\end{aligned}
$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies "to be a father means to be a parent", aj.,

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$
$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$
$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies "to be a father means to be a parent", aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: "Each person has at most two legs."

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **is a** relationships with the following semantics:

$$\begin{aligned} relation(X,Y) \wedge ako(Z,X) &\Rightarrow relation(Z,Y). \\ isa(X,Y) \wedge ako(Y,Z) &\Rightarrow isa(X,Z). \\ ako(X,Y) \wedge ako(Y,Z) &\Rightarrow ako(X,Z). \end{aligned}$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies "to be a father means to be a parent", aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: "Each person has at most two legs."
- Wordnet, Semantic Wiki, aj.

Frames

frame: Škoda Favorit
slots:
is a: car
has engine: four-stroke engine
has transmission system: manual
has carb: *value*: Jikov
default: Pierburg

- more structured than semantic networks

([MvL93])

Frames

frame: Škoda Favorit
slots:

- **is a**: car
- **has engine**: four-stroke engine
- **has transmission system**: manual
- **has carb**: *value*: Jikov
  default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

Frames

frame: Škoda Favorit
slots:
is a: car
has engine: four-stroke engine
has transmission system: manual
has carb: *value*: Jikov
default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.

Frames

frame: Škoda Favorit
slots:
- **is a**: car
- **has engine**: four-stroke engine
- **has transmission system**: manual
- **has carb**: *value*: Jikov
  default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.
- ☺ Facets allow non-monotonic reasoning.

Frames

frame: Škoda Favorit
slots:
  is a: car
  has engine: four-stroke engine
  has transmission system: manual
  has carb: *value*: Jikov
    default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.
- ☺ Facets allow non-monotonic reasoning.
- ☺ *Daemons* are triggers for actions perfomed on facets (read, write, delete). Can be used e.g for consistency checking.

Frames (2)

Example

Typically, `Škoda Favorit` **has carb** of type `Pierburg`, but this particular `Škoda Favorit` **has carb** of type `Jikov`.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]

Frames (2)

Example

Typically, `Škoda Favorit` **has carb** of type `Pierburg`, but this particular `Škoda Favorit` **has carb** of type `Jikov`.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - http://www.ai.sri.com/ okbc

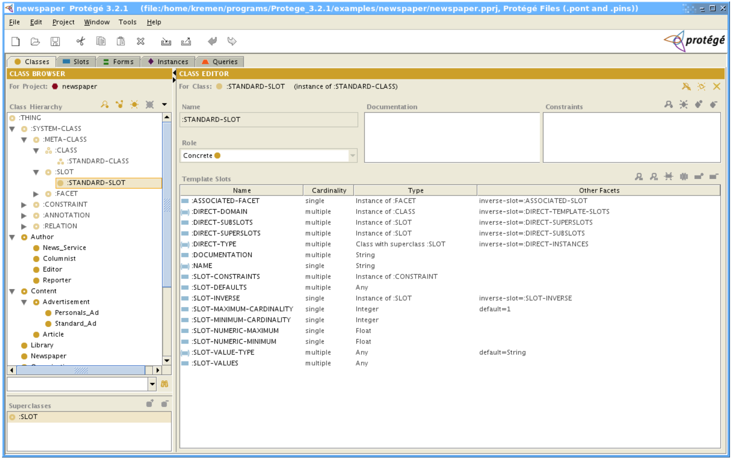
Frames (2)

Example

Typically, `Škoda Favorit` **has carb** of type `Pierburg`, but this particular `Škoda Favorit` **has carb** of type `Jikov`.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - http://www.ai.sri.com/ okbc
- Protégé - http://protege.stanford.edu/overview/protege-frames.html

Protégé


newspaper Protégé 3.2.1 (file:/home/kremen/programs/Protege_3.2.1/examples/newspaper/newspaper.pprj, Protégé Files (.pont and .pins))
File Edit Project Window Tools Help
protégé
Classes Slots Forms Instances Queries
CLASS BROWSER
For Project: newspaper
Class Hierarchy
:THING
:SYSTEM-CLASS
:META-CLASS
:CLASS
:STANDARD-CLASS
:SLOT
:STANDARD-SLOT
:FACET
:CONSTRAINT
:ANNOTATION
:RELATION
Author
News_Service
Columnist
Editor
Reporter
Content
Advertisement
Personals_Ad
Standard_Ad
Article
Library
Newspaper
Superclasses
:SLOT
CLASS EDITOR
For Class: :STANDARD-SLOT (instance of :STANDARD-CLASS)
Name
:STANDARD-SLOT
Documentation
Constraints
Role
Concrete
Template Slots
Name Cardinality Type Other Facets
:ASSOCIATED-FACET single Instance of :FACET inverse-slot=:ASSOCIATED-SLOT
:DIRECT-DOMAIN multiple Instance of :CLASS inverse-slot=:DIRECT-TEMPLATE-SLOTS
:DIRECT-SUBSLOTS multiple Instance of :SLOT inverse-slot=:DIRECT-SUPERSLOTS
:DIRECT-SUPERSLOTS multiple Instance of :SLOT inverse-slot=:DIRECT-SUBSLOTS
:DIRECT-TYPE multiple Class with superclass :SLOT inverse-slot=:DIRECT-INSTANCES
:DOCUMENTATION multiple String
:NAME single String
:SLOT-CONSTRAINTS multiple Instance of :CONSTRAINT
:SLOT-DEFAULTS multiple Any
:SLOT-INVERSE single Instance of :SLOT inverse-slot=:SLOT-INVERSE
:SLOT-MAXIMUM-CARDINALITY single Integer default=1
:SLOT-MINIMUM-CARDINALITY single Integer
:SLOT-NUMERIC-MAXIMUM single Float
:SLOT-NUMERIC-MINIMUM single Float
:SLOT-VALUE-TYPE multiple Any default=String
:SLOT-VALUES multiple Any

Frames and Semantics Networks – Summary

☺ very simple structures for knowledge representation,

Frames and Semantics Networks – Summary

- ☺ very simple structures for knowledge representation,
- ☺ nonmonotonic reasoning,

Frames and Semantics Networks – Summary

- ☺ very simple structures for knowledge representation,
- ☺ nonmonotonic reasoning,
- ☹ ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),

Frames and Semantics Networks – Summary

- ☺ very simple structures for knowledge representation,
- ☺ nonmonotonic reasoning,
- ☹ ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- ☹ problems – querying, debugging.

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.

Languages sketched so far aren't enough ?

- Relational algebra

Languages sketched so far aren't enough ?

- Relational algebra
  - accepts CWA and supports just *finite domains*.

Languages sketched so far aren't enough ?

- Relational algebra
  - accepts CWA and supports just *finite domains*.
- Semantic networks and Frames

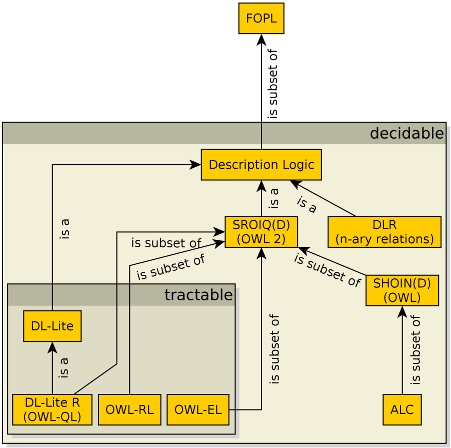
Languages sketched so far aren't enough ?

- Relational algebra
  - accepts CWA and supports just *finite domains*.
- Semantic networks and Frames
  - Lack well defined (declarative) semantics

Languages sketched so far aren't enough ?

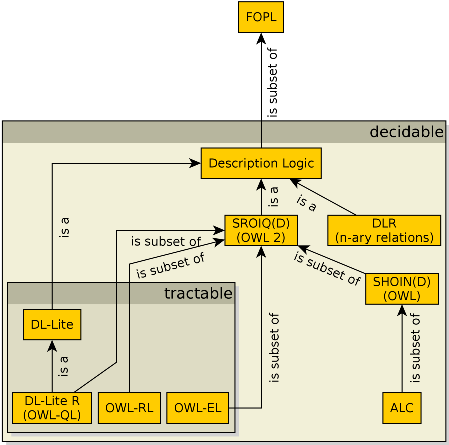
- Relational algebra
  - accepts CWA and supports just *finite domains*.
- Semantic networks and Frames
  - Lack well defined (declarative) semantics
  - What is the semantics of a "slot" in a frame (relation in semantic networks) ? The slot **must/might** be filled **once/multiple times** ?

What are Description Logics ?



What are Description Logics ?

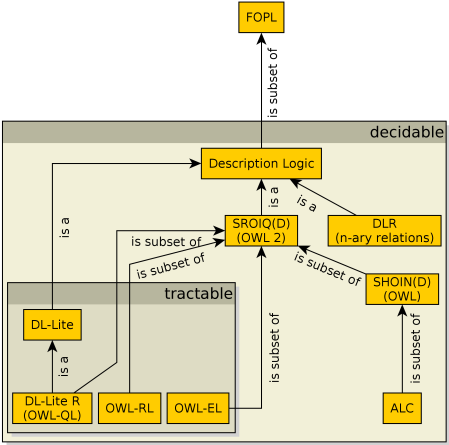
Description logics (DLs) are (almost exclusively) decidable subsets of FOPL aimed at modeling *terminological incomplete knowledge*.



What are Description Logics ?

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

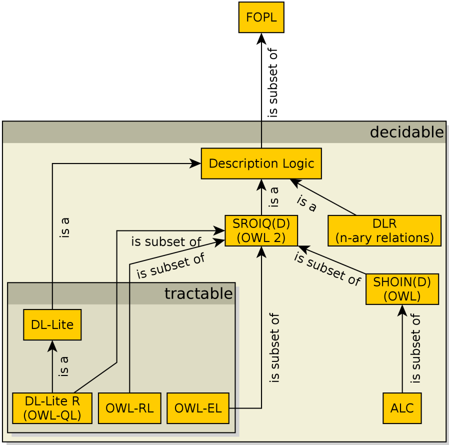
- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.



What are Description Logics ?

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

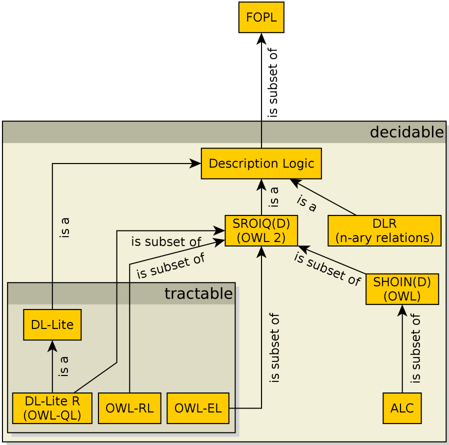
- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.
- 90's $\mathcal{ALC}$



What are Description Logics ?

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

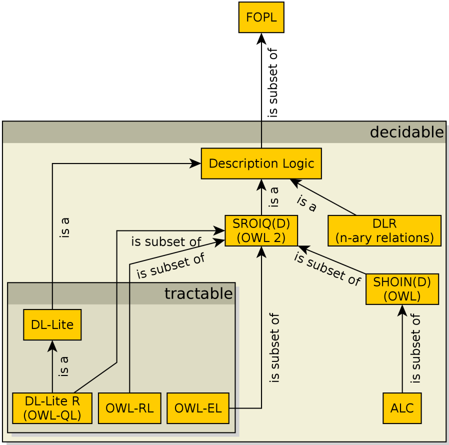
- first languages emerged as an experiment of giving formal semantics to semantic networks and frames. First implementations in 80's – KL-ONE, KAON, Classic.
- 90's $\mathcal{ALC}$
- 2004 $\mathcal{SHOIN}(\mathcal{D})$ – OWL



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 $\mathcal{K}$ (in OWL refered as Ontology) of DLs 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 $\mathcal{K}$ (in OWL refered as Ontology) of DLs 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 $\mathcal{K}$ (in OWL refered as Ontology) of DLs 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. $\mathcal{A} = \{Man(JOHN)\}$

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 $\mathcal{K}$ (in OWL refered as Ontology) of DLs 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. $\mathcal{A} = \{Man(JOHN)\}$

- 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)
$\forall R \cdot C$	$\{a \mid \forall b((a, b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}})\}$	(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)
$\forall R \cdot C$	$\{a \mid \forall b\,((a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}})\}$	(universal restriction)
$\exists R \cdot C$	$\{a \mid \exists b\,((a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}})\}$	(existential restriction)

	axiom	$\mathcal{I} \models$ axiom iff	*description*
TBOX	$C_1 \sqsubseteq C_2$	$C_1{}^{\mathcal{I}} \subseteq C_2{}^{\mathcal{I}}$	(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)
$\forall R \cdot C$	$\{a \mid \forall b\,((a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}})\}$	(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 $S = \mathcal{T} \cup \mathcal{A}$), then

Logical Consequence

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

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 $S = \mathcal{T} \cup \mathcal{A}$), then

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 $S = \mathcal{T} \cup \mathcal{A}$), then

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 $S = \mathcal{T} \cup \mathcal{A}$), then

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*.

- How to express a set of persons that have just men as their descendants, if any ?

$\mathcal{ALC}$ – 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 $hasChild$.

- How to express a set of persons that have just men as their descendants, if any ?
  - $Person \sqcap \forall hasChild \cdot Man$

$\mathcal{ALC}$ – 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*.

- How to express a set of persons that have just men as their descendants, if any ?
  - $Person \sqcap \forall hasChild \cdot Man$
- How to define concept *GrandParent* ?

$\mathcal{ALC}$ – 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 $hasChild$.

- How to express a set of persons that have just men as their descendants, if any ?
  - $Person \sqcap \forall hasChild \cdot Man$
- How to define concept $GrandParent$?
  - $GrandParent \equiv Person \sqcap \exists hasChild \cdot \exists hasChild \cdot \top$

$\mathcal{ALC}$ – 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*.

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

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

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

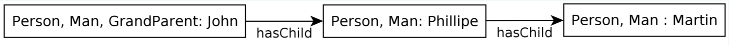
Example

- 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

Every satisfiable ALC concept[a] C has a model in the shape of a *rooted tree*.

[a] Concept is satisfiable, if at least one model interprets it as a non-empty set

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property

Every satisfiable ALC concept[a] C has a model in the shape of a *rooted tree*.

[a] Concept is satisfiable, if at least one model interprets it as a non-empty set

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property

Every satisfiable ALC concept[a] C has a model in the shape of a *rooted tree*.

[a] Concept is satisfiable, if at least one model interprets it as a non-empty set

Finite model property

Every consistent theory $\mathcal{K}$ has a *finite model*.

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property

Every satisfiable ALC concept[a] C has a model in the shape of a *rooted tree*.

[a] Concept is satisfiable, if at least one model interprets it as a non-empty set

Finite model property

Every consistent theory $\mathcal{K}$ has a *finite model*.

Both properties represent important characteristics of a DL that directly influence inferencing (see next lecture).

Shape of DL Models

The last example revealed several important properties of DL models:

Tree model property

Every satisfiable ALC concept[a] C has a model in the shape of a *rooted tree*.

[a] Concept is satisfiable, if at least one model interprets it as a non-empty set

Finite model property

Every consistent theory $\mathcal{K}$ has a *finite model*.

Both properties represent important characteristics of a DL that directly influence inferencing (see next lecture).

In particular (generalized) TMP is a characteristics that is shared by most DLs and significantly reduces their computational complexity.

Example

Example

primitive concept

defined concept

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

Example – CWA $\times$ 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*)
¬*Patricide*(*THERSANDROS*)

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

Example – CWA × OWA

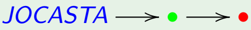
Example

ABOX

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

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

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



Example – CWA × OWA

Example

ABOX

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

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

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

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

JOCASTA ⟶ • ⟶ •

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 ⟶ • ⟶ x

* Vladimír Mařík, Olga Štěpánková, and Jiří Lažanský.
Umělá inteligence 6 [in czech], Chapter "Ontologie a deskripční logiky".
Academia, 2013.

Vladimír Mařík, Olga Štěpánková, and Jiří Lažanský.
Umělá inteligence 1.
Academia, 1993.

* 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.

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