Description logics – syntax, semantics, reasoning, querying and debugging

Petr Křemen

petr.kremen@fel.cvut.cz

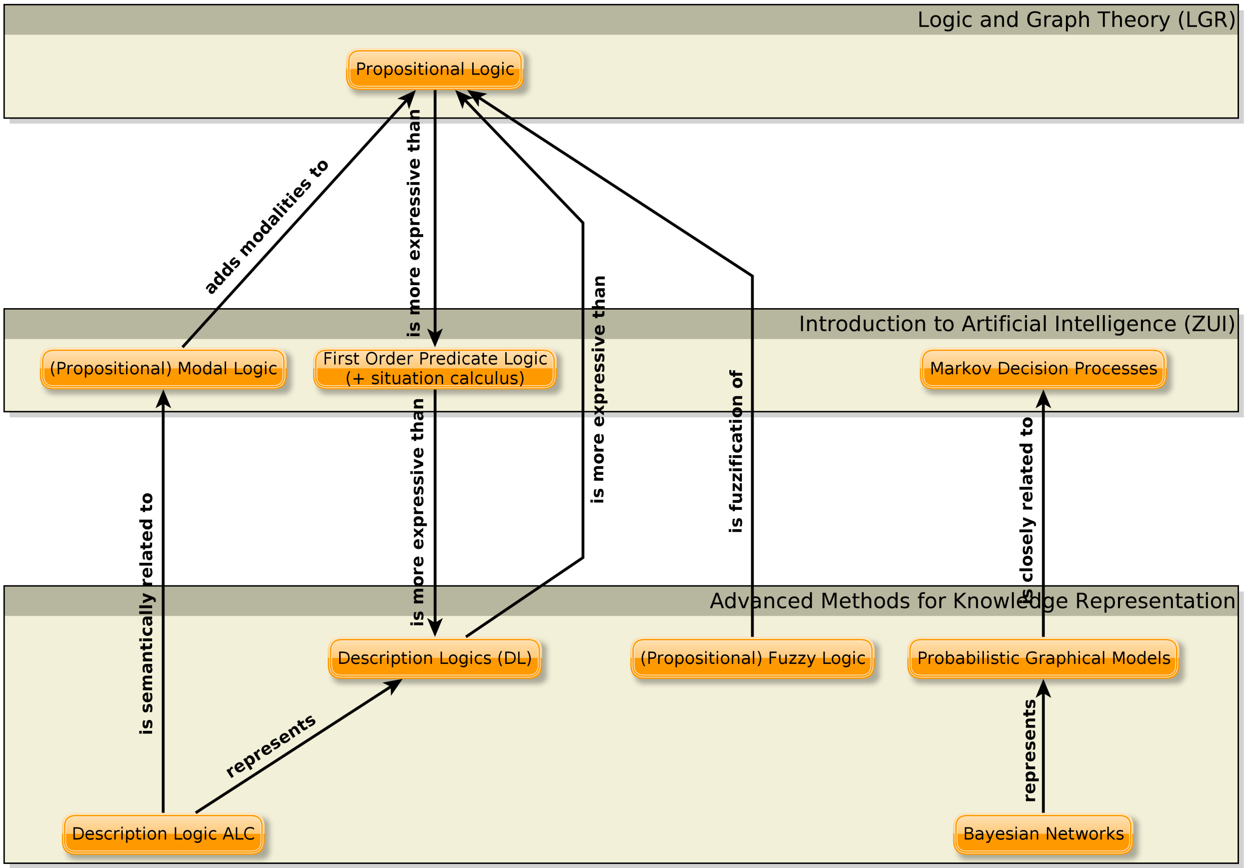
October 26, 2015

1 Course Information

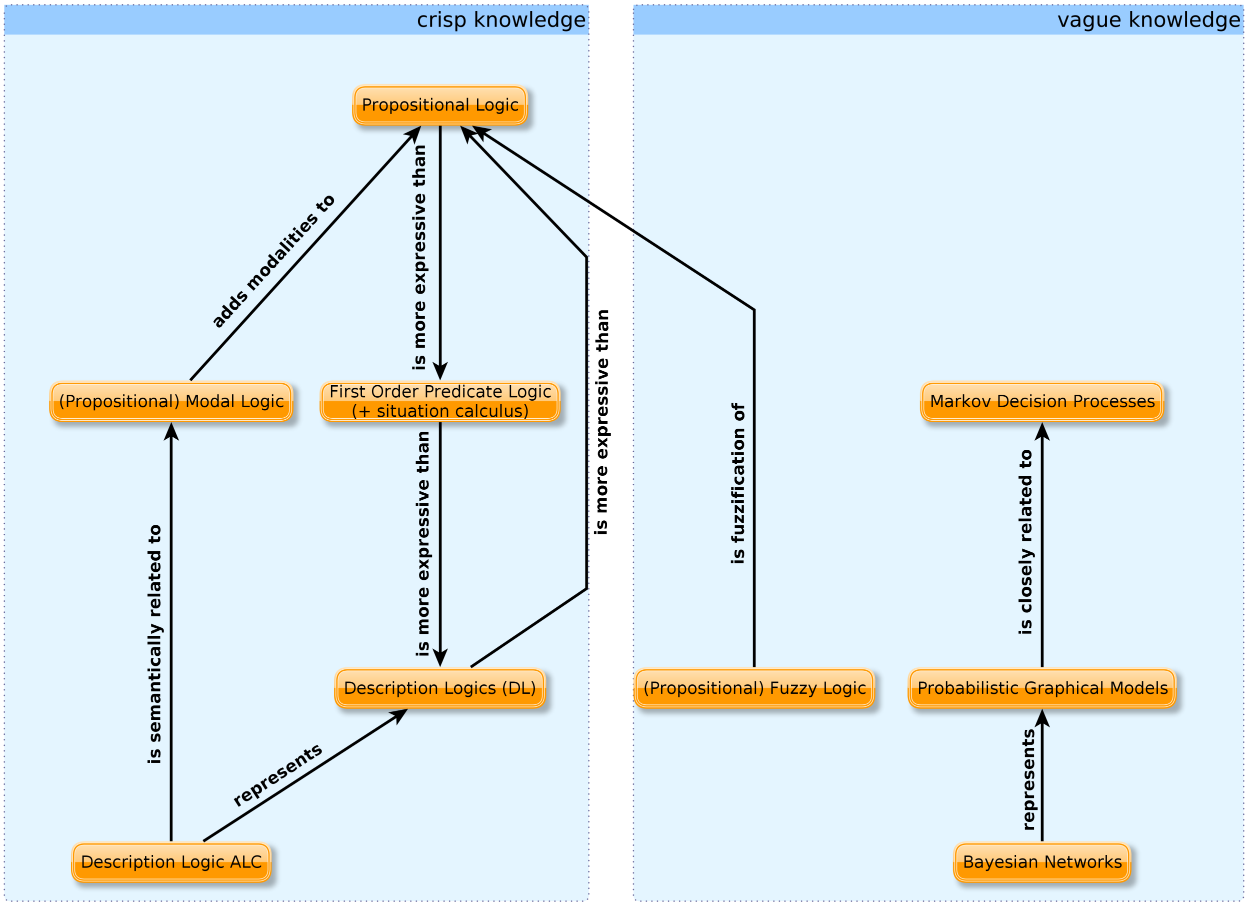
Course Information

- 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 !!!**

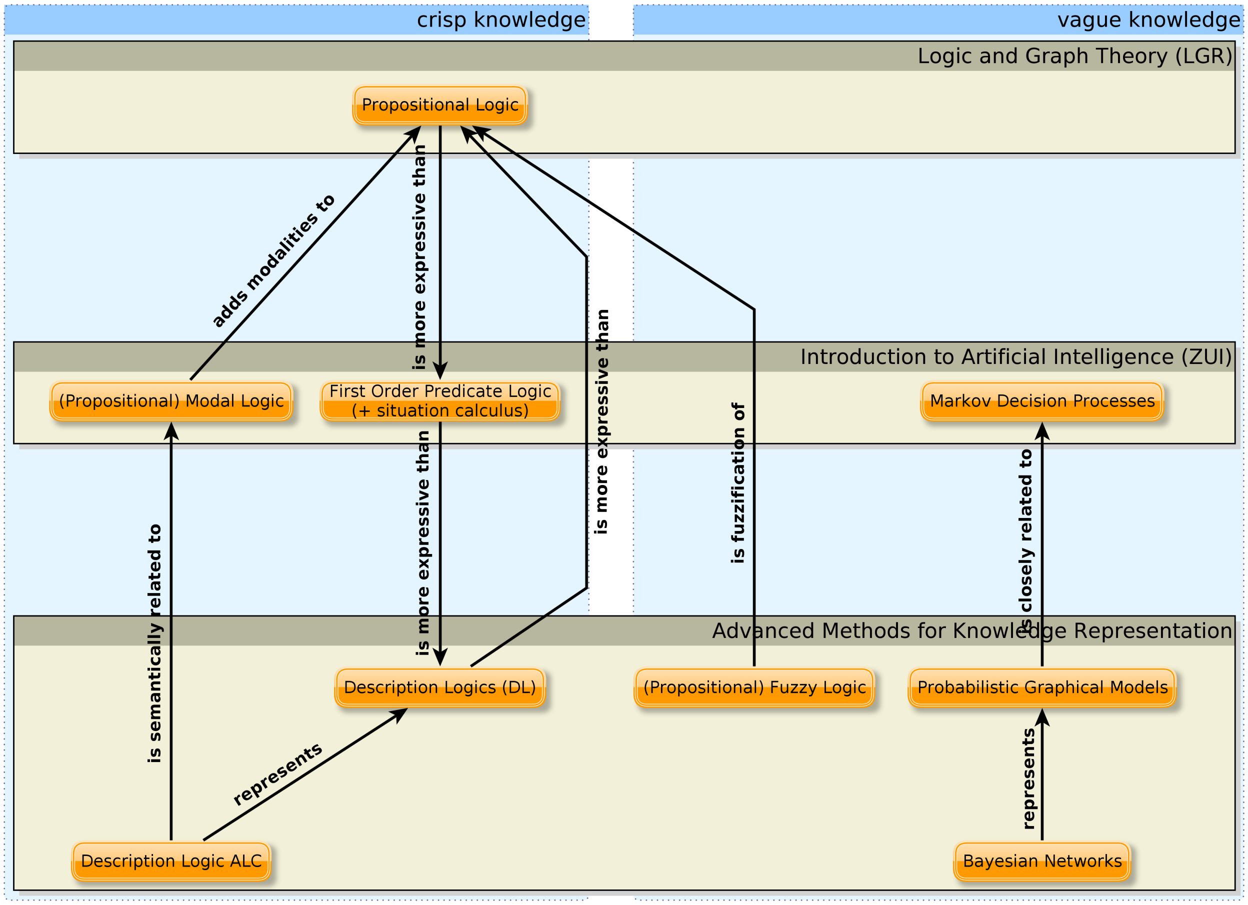
Technical Roadmap


Logic and Graph Theory (LGR)
Propositional Logic
adds modalities to
is more expressive than
is more expressive than
is fuzzification of
Introduction to Artificial Intelligence (ZUI)
(Propositional) Modal Logic
First Order Predicate Logic (+ situation calculus)
Markov Decision Processes
is semantically related to
is more expressive than
is closely related to
Advanced Methods for Knowledge Representation
Description Logics (DL)
(Propositional) Fuzzy Logic
Probabilistic Graphical Models
represents
represents
Description Logic ALC
Bayesian Networks

Technical Roadmap (2)

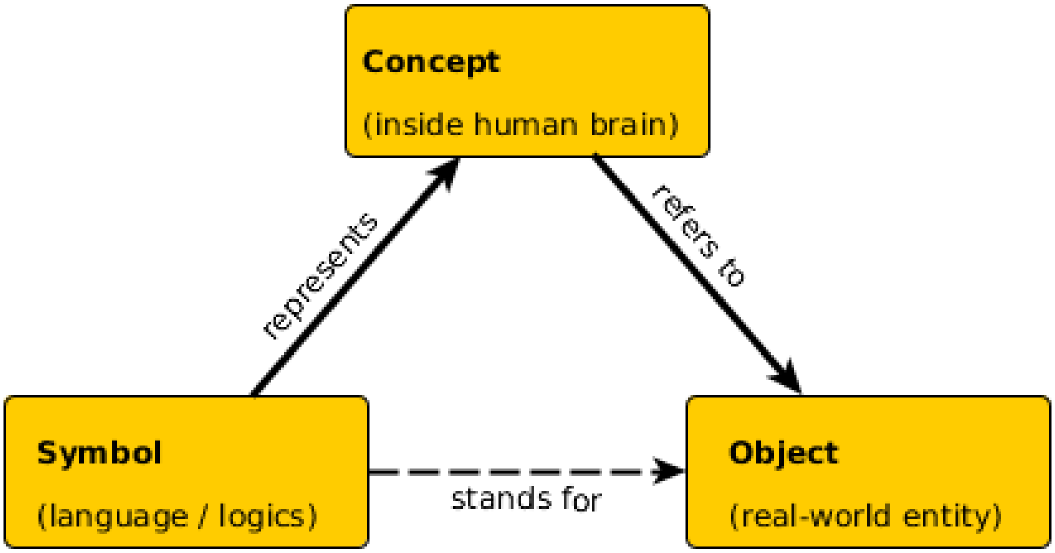


Technical Roadmap (3)


crisp knowledge
vague knowledge
Logic and Graph Theory (LGR)
Propositional Logic
adds modalities to
is more expressive than
is more expressive than
is fuzzification of
Introduction to Artificial Intelligence (ZUI)
(Propositional) Modal Logic
First Order Predicate Logic
(+ situation calculus)
Markov Decision Processes
is semantically related to
is more expressive than
is closely related to
Advanced Methods for Knowledge Representation
Description Logics (DL)
(Propositional) Fuzzy Logic
Probabilistic Graphical Models
represents
represents
Description Logic ALC
Bayesian Networks

2 Towards Description Logics

Semiotic Triangle



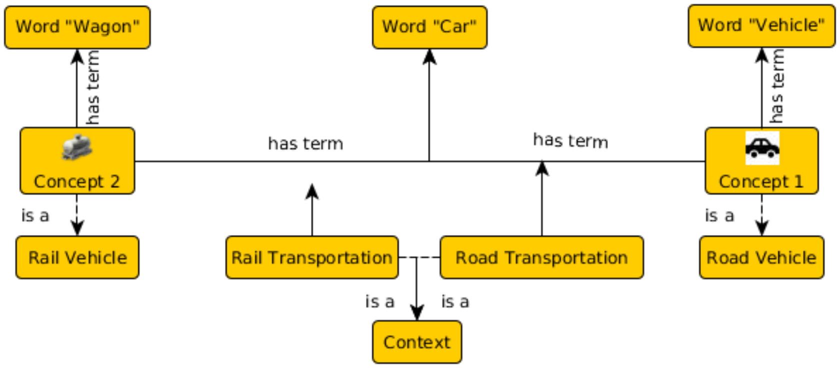
refers to ∼ modeled by *ontologies*; you can learn in AE0M33OSW course

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

Ontologies

Ontologies
in IT are formal informational artifacts *explicitely representing shared conceptualization.*

- basic idea = we need to model (and reason) on *concepts* (i.e. "meanings") not terms (i.e. "symbols", "words", "phrases"). We need to know, what our mean.
- compare words *Man* vs. *Person.*
- but we need to use words to model the concepts ...



Ontologies (2)

Principle of application. A concept satisfies the principle of application if we are able to decide, whether an "object" belongs to the concept or not. E.g. *nice* vs. *red* vs. *woman.*

Principle of identity. Each concept is equipped with a principle of identity saying, what must be fulfilled for an object to belong to the concept. E.g. *an artificial key, birth number* vs. *particular human brain*

Many concept types – universals vs. individuals, endurants vs. perdurants, etc.

... and much more

Ontologies can be represented formally, in order to exploit *automated reasoning on concepts/meanings.*

3 Logics

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

  Example 1. “John is clever.″ $\Rightarrow \neg$“John fails at exam.″
- first order predicate logic

  Example 2. $(\forall x)(Clever(x) \Rightarrow \neg((\exists y)(Exam(y) \wedge Fails(x,y))))$.
- (propositional) modal logic

  Example 3. $\Box((\forall x)(Clever(x) \Rightarrow \Diamond\neg((\exists y)(Exam(y) \wedge Fails(x,y)))))$.
- ... what is the meaning of these formulas ?

Logics for 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 4. How to check satisfiability of the formula $A \lor (\neg(B \land A) \lor B \land C)$?

syntax – atomic formulas and $\neg$, $\land$, $\lor$, $\Rightarrow$

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 5. What is the meaning of this sentence ?

$$\begin{aligned}&(\forall x_1)((Student(x_1) \land (\exists x_2)(GraduateCourse(x_2) \land isEnrolledTo(x_1, x_2)))\\ &\Rightarrow (\forall x_3)(isEnrolledTo(x_1, x_3) \Rightarrow GraduateCourse(x_3)))\end{aligned}$$

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

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 $\neg$, $\land$, $\lor$, $\Rightarrow$, $\forall$,$\exists$)

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

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.

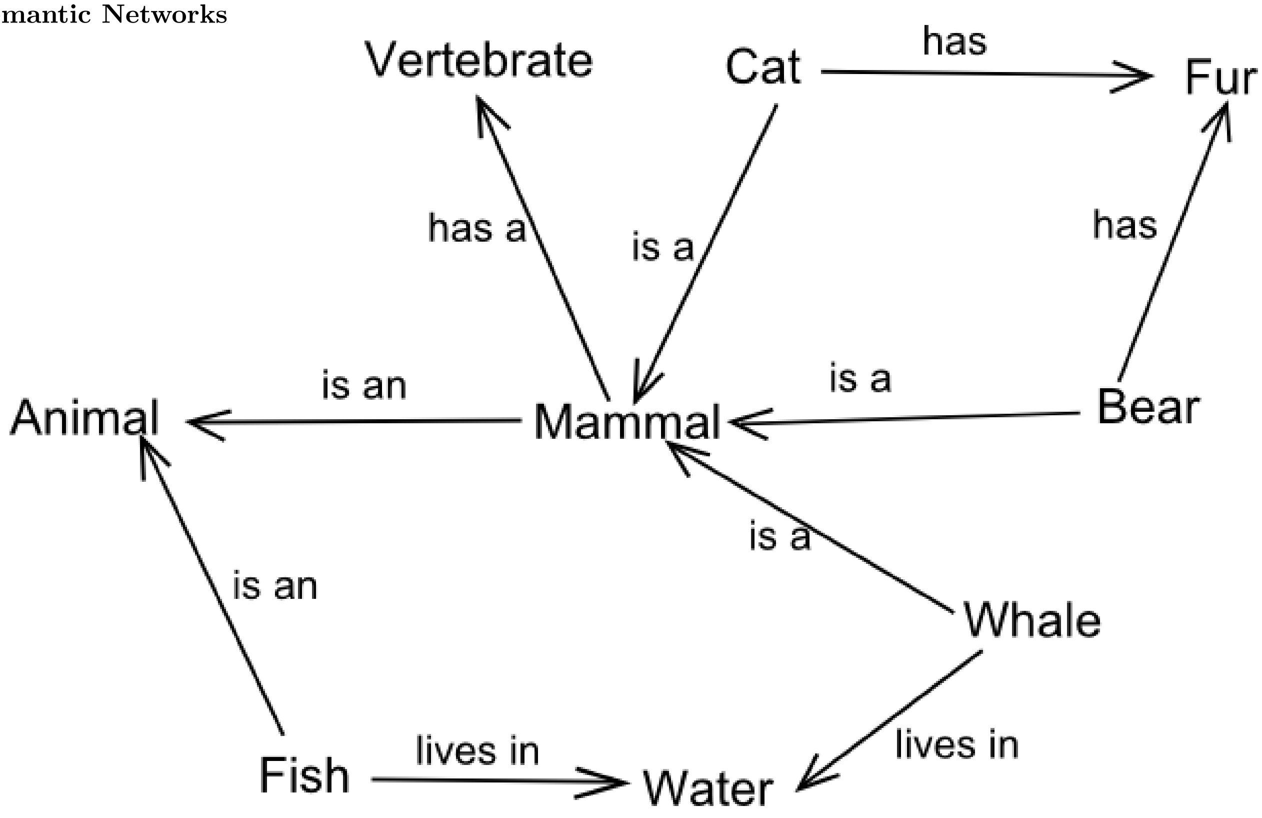
As a result, FOPL is *monotonic*, i.e.

monotonicity
No conclusion can be invalidated by adding extra knowledge.

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

4 Semantic Networks and Frames

Semantic Networks



(©wikipedia.org)

Nodes = entities (individuals, classes),

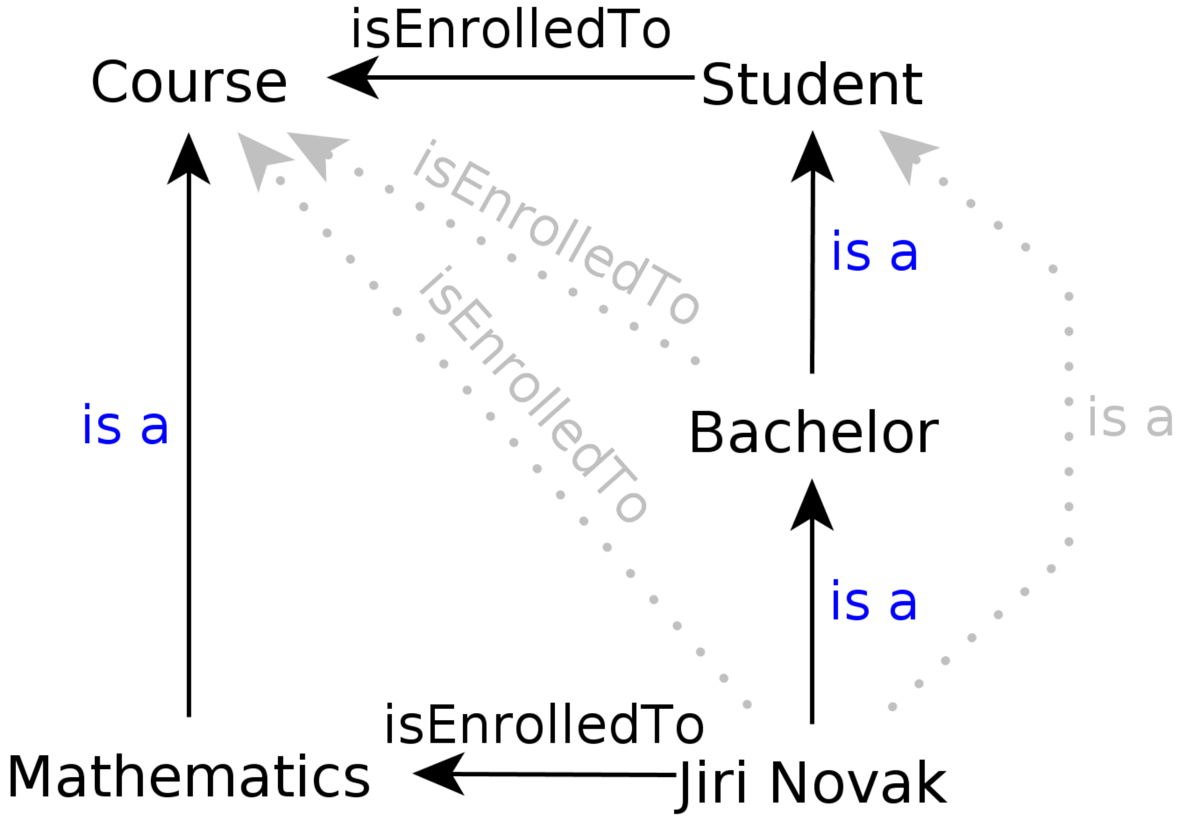
Edges = binary relations

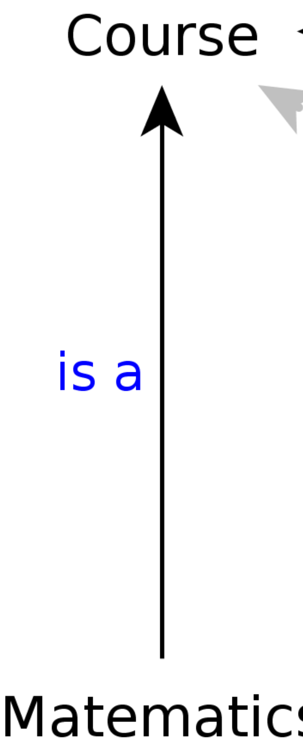
- The only possible inference is *inheritance* by means of **is a** relationship.

  Example

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

Semantic Networks (2)


isEnrolledTo
Course
Student
isEnrolledTo
isEnrolledTo
is a
is a
is a
Bachelor
is a
is a
isEnrolledTo
Mathematics
Jiri Novak



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:

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

- ☹ 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", etc.,
- ☹ no way to express more complex constructs, like cardinality restrictions: "Each person has at most two legs."
- Wordnet, Semantic Wiki, 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.

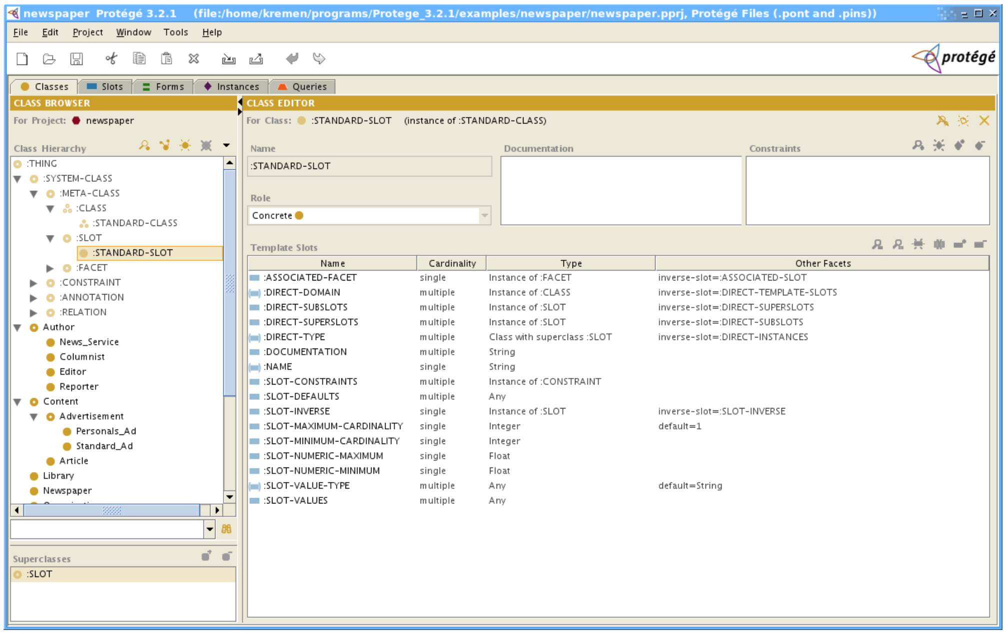
☺ *Daemons* are triggers for actions perfomed on facets (read, write, delete). Can be used e.g for consistency checking.

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]
- OKBC - http://www.ai.sri.com/ okbc
- Protégé - http://protege.stanford.edu/overview/protege-frames.html

Protégé



For Project: newspaper

For Class: :STANDARD-SLOT (instance of :STANDARD-CLASS)

Name: :STANDARD-SLOT

Role: Concrete

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

5 Towards Description Logics

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.

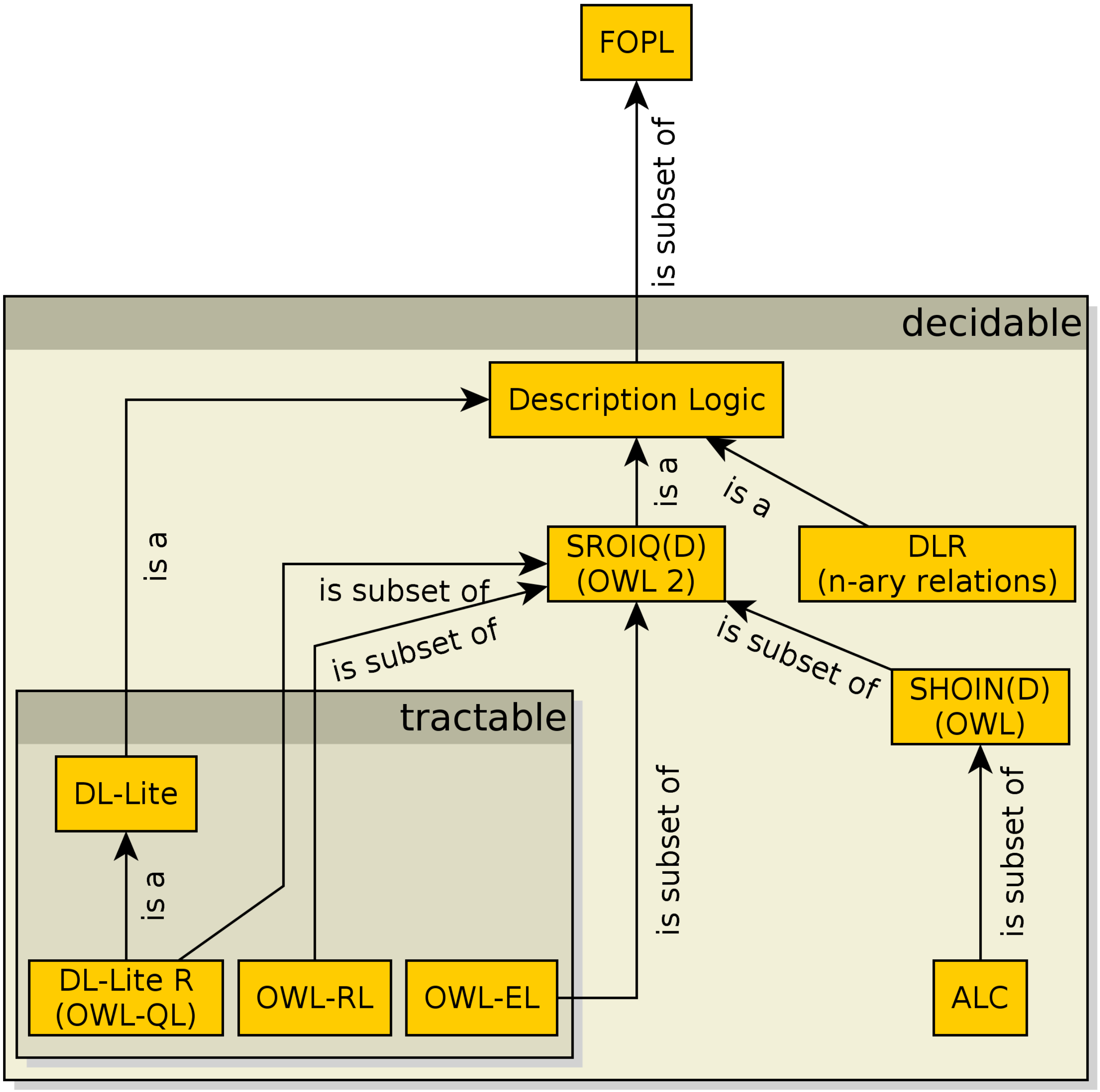
Languages sketched so far aren't enough ?

- Relational algebra
  - 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 ?

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



6 $\mathcal{ALC}$ Language

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):
- **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)

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)

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

[1] two different individuals denote two different domain elements

$\mathcal{ALC}$ – 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, $\mathit{GrandParent}$ and atomic role $\mathit{hasChild}$.

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

$$\begin{aligned}\forall x \, (\mathit{GrandParent}(x) \equiv (\mathit{Person}(x) \wedge \exists y \, (\mathit{hasChild}(x,y) \\ \wedge \exists z \, (\mathit{hasChild}(y,z)))))\end{aligned}$$

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

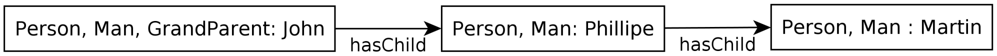
Example

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

Interpretation – Example

Example

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

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*.

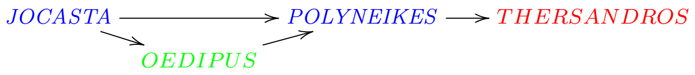
Both properties represent important characteristics of $\mathcal{ALC}$ that significantly speed-up reasoning.

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

Example – CWA × OWA

Example 6. **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 $(\exists hasChild \cdot (Patricide \sqcap \exists hasChild \cdot \neg Patricide))(JOCASTA)$,

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

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 ?

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

7 Inference Problems

Inference Problems in TBOX

We have introduced syntax and semantics of the language $\mathcal{ALC}$. Now, let's look on automated reasoning. Having a $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For TBOX $\mathcal{T}$ and concepts $C_{(i)}$, we want to decide whether

(unsatisfiability) concept C is *unsatisfiable*, i.e. $\mathcal{T} \models C \sqsubseteq \bot$?

(subsumption) concept C_1 *subsumes* concept C_2, i.e. $\mathcal{T} \models C_2 \sqsubseteq C_1$?

(equivalence) two concepts C_1 and C_2 are *equivalent*, i.e. $\mathcal{T} \models C_1 \equiv C_2$?

(disjoint) two concepts C_1 and C_2 are *disjoint*, i.e. $\mathcal{T} \models C_1 \sqcap C_2 \sqsubseteq \bot$?

All these tasks can be reduced to unsatisfiability checking of a single concept ...

Reducting Subsumption to Unsatisfiability

Example
These reductions are straighforward – let's show, how to reduce subsumption checking to unsatisfiability checking. Reduction of other inference problems to unsatisfiability is analogous.

$$\begin{array}{rcl}
(\mathcal{T} \models C_1 \sqsubseteq C_2) & & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & \mathcal{I} \models C_1 \sqsubseteq C_2) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & {C_1}^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & {C_1}^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus {C_2}^{\mathcal{I}}) \subseteq \emptyset & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & \mathcal{I} \models C_1 \sqcap \neg C_2 \sqsubseteq \bot & \text{iff} \\
(\mathcal{T} \models C_1 \sqcap \neg C_2 \sqsubseteq \bot) & &
\end{array}$$

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a_1, a_2)$?

(instance retrieval) find all individuals a, for which $\mathcal{T} \cup \mathcal{A} \models C(a)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Reduction of concept unsatisfiability to theory consistency

Example
Consider an $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a concept C and a fresh individual a_f not occuring in $\mathcal{K}$:

$$\begin{array}{rl}
(\mathcal{T} \models C \sqsubseteq \bot) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies \mathcal{I} \models C \sqsubseteq \bot) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies C^{\mathcal{I}} \subseteq \emptyset) & \text{iff} \\
\neg \left[(\exists \mathcal{I})(\mathcal{I} \models \mathcal{T} \wedge C^{\mathcal{I}} \not\subseteq \emptyset)\right] & \text{iff} \\
\neg \left[(\exists \mathcal{I})(\mathcal{I} \models \mathcal{T} \wedge {a_f}^{\mathcal{I}} \in C^{\mathcal{I}})\right] & \text{iff} \\
(\mathcal{T}, \{C(a_f)\}) \quad \text{is inconsistent} &
\end{array}$$

Note that for more expressive description logics than $\mathcal{ALC}$, the ABOX has to be taken into account as well due to its interaction with TBOX.

8 Inference Algorithms

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [BCM+03].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite

other ... – e.g. resolution-based transformation to finite automata , etc.

We will introduce tableau algorithms.

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking theory consistency in a simple manner: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$." (resp. theory $\mathcal{K}$)**
- Each TA can be seen as a *production system* :
  - *state* of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - *inference rules* ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists, \sqcap$, etc. and serve to modify the completion graphs according to
  - choosen *strategy* for rule application
- TAs are not new in DL – they were known for FOL as well.

Completion Graphs

completion graph is a labeled oriented graph $G = (V_G, E_G, L_G))$, where each node $x \in V_G$ is labeled with a set $L_G(x)$ of concepts and each edge $\langle x, y \rangle \in E_G$ is labeled with a set of edges $L_G(\langle x, y \rangle)$[2]

direct clash occurs in a completion graph $G = (V_G, E_G, L_G))$, if $\{A, \neg A\} \subseteq L_G(x)$, or $\bot \in L_G(x)$, for some atomic concept A and a node $x \in V_G$

complete completion graph is a completion graph $G = (V_G, E_G, L_G))$, to which no completion rule from the set of TA completion rules can be applied.

Do not mix with notion of *complete graphs* known from graph theory.

Completion Graphs (2)

We define also $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, where $\mathcal{A}_G$ is an ABOX constructed from G, as follows

- $C(a)$ for each node $a \in V_G$ and each concept $C \in L_G(a)$ and
- $R(a_1, a_2)$ for each edge $\langle a_1, a_2 \rangle \in E_G$ and each role $R \in L_G(a_1, a_2)$

8.1 Tableau Algorithm for $\mathcal{ALC}$

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0 (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(C_1 \sqcap C_2)$ is equivalent (de Morgan rules) to $\neg C_1 \sqcup \neg C_2$).

1 (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:

[2] Next in the text the notation is often shortened as $L_G(x, y)$ instead of $L_G(\langle x, y \rangle)$.

- for each $C(a) \in \mathcal{A}$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
- for each $R(a_1, a_2) \in \mathcal{A}$ put $\langle a_1, a_2\rangle \in E_{G_0}$ and $R \in L_{G_0}(a_1, a_2)$
- Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties.

Tableau algorithm for $\mathcal{ALC}$ without TBOX (2)

. . .

2 (Consistency Check) Current algorithm state is S. If each $G \in S$ contains a direct clash, terminate with result "INCONSISTENT"

3 (Model Check) Let's choose one $G \in S$ that doesn't contain a direct clash. If G is complete w.r.t. rules shown next, the algorithm terminates with result "CONSISTENT"

4 (Rule Application) Find a rule that is applicable to G and apply it. As a result, we obtain from the state S a new state S'. Jump to step 2.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a_1)$ and there exists no $a_2 \in V_G$ such that $R \in L_G(a_1, a_2)$ and at the same time $C \in L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{a_2\}, E_G \cup \{\langle a_1, a_2\rangle\}, L_{G'})$, a $L_{G'}(a_2) = \{C\}$, $L_{G'}(a_1, a_2) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

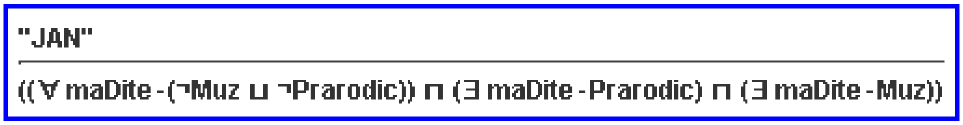
if $(\forall R \cdot C) \in L_G(a_1)$ and there exists $a_2 \in V_G$ such that $R \in L_G(a, a_1)$ and at the same time $C \notin L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a_2) = L_G(a_2) \cup \{C\}$ and otherwise is the same as L_G.

TA Run Example

Example 7. Let's check consistency of the ontology $\mathcal{K}_2 = (\emptyset, \mathcal{A}_2)$, where $\mathcal{A}_2 = \{(\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \neg\exists maDite \cdot (Muz \sqcap Prarodic))(JAN)\}$.

- Let's transform the concept into NNF: $\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \forall maDite \cdot (\neg Muz \sqcup \neg Prarodic)$
- Initial state G_0 of the TA is



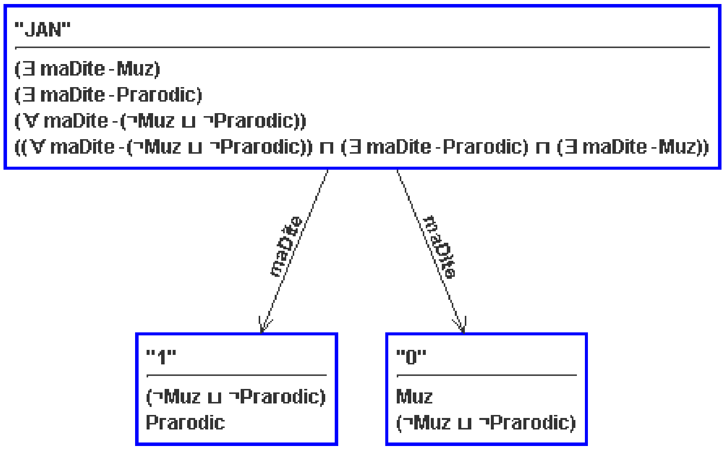
"JAN"

((∀ maDite-(¬Muz ⊔ ¬Prarodic)) ⊓ (∃ maDite-Prarodic) ⊓ (∃ maDite-Muz))

TA Run Example (2)

Example 8. …

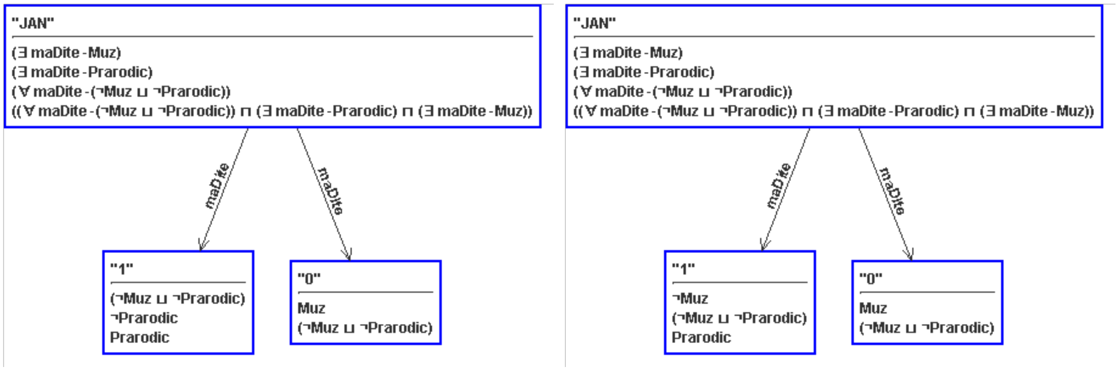
- Now, four sequences of steps 2,3,4 of the TA are performed. TA state in step 4, evolves as follows:
- $\{G_0\} \xrightarrow{\sqcap\text{-rule}} \{G_1\} \xrightarrow{\exists\text{-rule}} \{G_2\} \xrightarrow{\exists\text{-rule}} \{G_3\} \xrightarrow{\forall\text{-rule}} \{G_4\}$, where G_4 is


"JAN"
(∃ maDite - Muz)
(∃ maDite - Prarodic)
(∀ maDite - (¬Muz ⊔ ¬Prarodic))
((∀ maDite - (¬Muz ⊔ ¬Prarodic)) ⊓ (∃ maDite - Prarodic) ⊓ (∃ maDite - Muz))
maDite
maDite
"1"
(¬Muz ⊔ ¬Prarodic)
Prarodic
"0"
Muz
(¬Muz ⊔ ¬Prarodic)

TA Run Example (3)

Example 9. …

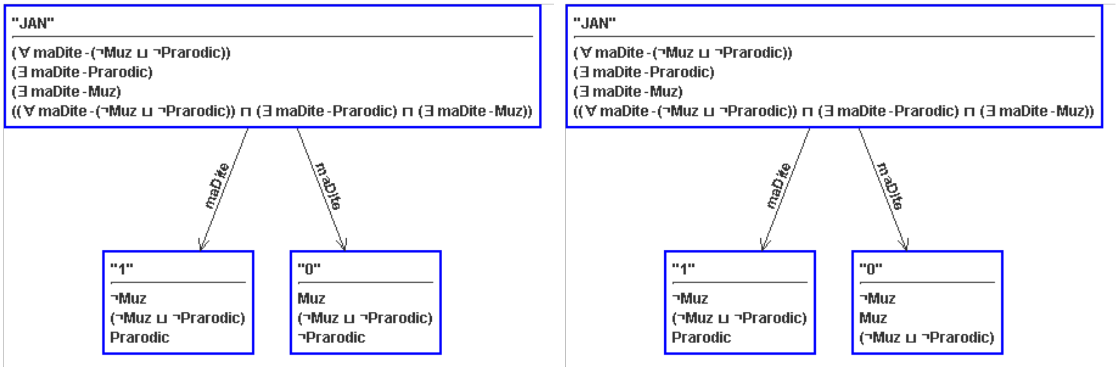
- By now, we applied just deterministic rules (we still have just a single completion graph). At this point no other deterministic rule is applicable.
- Now, we have to apply the ⊔-rule to the concept $\neg Muz \sqcup \neg Rodic$ either in the label of node "0", or in the label of node "1". Its application e.g. to node "1" we obtain the state $\{G_5, G_6\}$ (G_5 left, G_6 right)



TA Run Example (4)

Example 10. …

- We see that G_5 contains a direct clash in node "1". The only other option is to go through the graph G_6. By application of ⊔-rule we obtain the state $\{G_5, G_7, G_8\}$, where G_7 (left), G_8 (right) are derived from G_6 :



- G_7 is complete and without direct clash.

TA Run Example (5)

Example 11. ... A canonical model $\mathcal{I}_2$ can be created from G_7. Is it the only model of $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $\text{“}JAN''^{\mathcal{I}_2} = Jan$, $\text{“}0''^{\mathcal{I}_2} = i_2$, $\text{“}1''^{\mathcal{I}_2} = i_1$,

Finiteness

Finiteness of the TA is an easy consequence of the following:

- $\mathcal{K}$ is finite
- in each step, TA state can be enriched at most by one completion graph (only by application of $\rightarrow_\sqcup$ rule). Number of disjunctions ($\sqcup$) in $\mathcal{K}$ is finite, i.e. the $\sqcup$ can be applied just finite number of times.
- for each completion graph $G = (V_G, E_G, L_G)$ it holds that number of nodes in V_G is less or equal to the number of individuals in $\mathcal{A}$ plus number of existential quantifiers in $\mathcal{A}$.
- after application of any of the following rules $\rightarrow_\sqcap$, $\rightarrow_\exists$, $\rightarrow_\forall$ graph G is either enriched with a new node, new edge, or labeling of an existing node/edge is enriched. All these operations are finite.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
  - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.
  - As a result $a_1{}^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.
  - Next, $i \in \Delta^{\mathcal{I}}$ must exist such that $\langle a_1{}^{\mathcal{I}}, i\rangle \in R^{\mathcal{I}}$ and at the same time $i \in C^{\mathcal{I}}$.
  - By application of $\rightarrow_\exists$ a new node a_2 was created in G_{i+1} and the label of edge $\langle a_1, a_2\rangle$ and node a_2 has been adjusted.

  - It is enough to place $i = {a_2}^{\mathcal{I}}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.

- For other rules, the soundness is shown in a similar way.

Completeness

- To prove completeness of the TA, it is necessary to construct a model for each complete completion graph G that doesn't contain a direct clash. Canonical model $\mathcal{I}$ can be constructed as follows:
  - the domain $\Delta^{\mathcal{I}}$ will consist of all nodes of G.
  - for each atomic concept A let's define $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
  - for each atomic role R let's define $R^{\mathcal{I}} = \{\langle a_1, a_2\rangle \mid R \in L_G(a_1, a_2)\}$
- Observe that $\mathcal{I}$ is a model of $\mathcal{A}_G$. A backward induction can be used to show that $\mathcal{I}$ must be also a model of each previous step and thus also $\mathcal{A}$.

A few remarks on TAs

- Why we need completion graphs ? Aren't ABOXes enough to maintain the state for TA ?
  - indeed, for $\mathcal{ALC}$ they would be enough. However, for complex DLs a TA state cannot be stored in an ABOX.
- What about complexity of the algorithm ?
  - P-SPACE (between NP and EXP-TIME).

General Inclusions

We have presented the tableau algorithm for consistency checking of $\mathcal{K} = (\emptyset, \mathcal{A})$. How the situation changes when $\mathcal{T} \neq \emptyset$?

- consider $\mathcal{T}$ containing axioms of the form $C_i \sqsubseteq D_i$ for $1 \leq i \leq n$. Such $\mathcal{T}$ can be transformed into a single axiom

$$\top \sqsubseteq \top_C$$

  where $\top_C$ denotes a concept $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$

- for each model $\mathcal{I}$ of the theory $\mathcal{K}$, each element of $\Delta^{\mathcal{I}}$ must belong to $\top_C^{\mathcal{I}}$. How to achieve this ?

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

if $\top_C \notin L_G(a)$ for some $a \in V_G$.

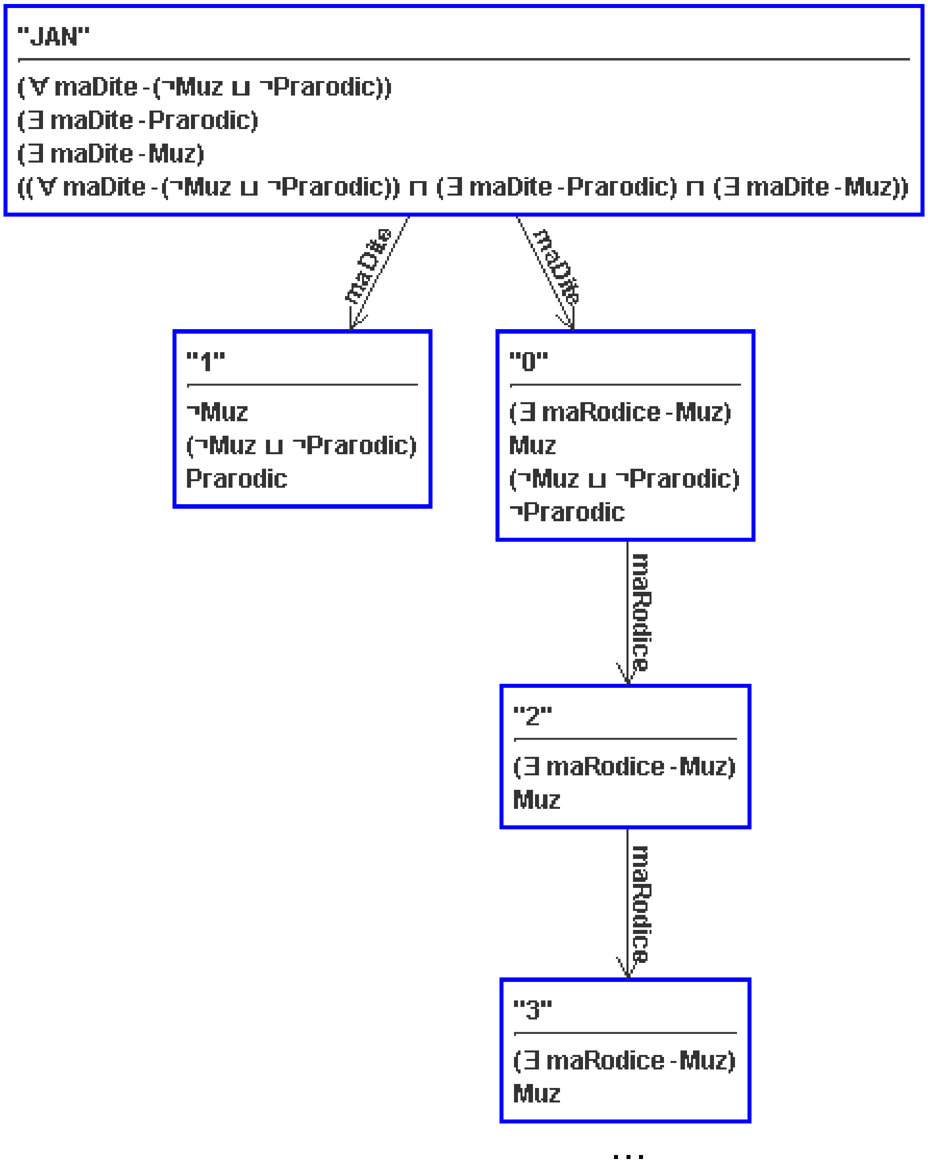
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ and otherwise is the same as L_G.

Example

Consider $\mathcal{K}_3 = (\{Muz \sqsubseteq \exists maRodice \cdot Muz\}, \mathcal{A}_2)$. Then $\top_C$ is $\neg Muz \sqcup \exists maRodice \cdot Muz$. Let's use the introduced TA enriched by $\rightarrow_{\sqsubseteq}$ rule. Repeating several times the application of rules $\rightarrow_{\sqsubseteq}$, $\rightarrow_{\sqcup}$, $\rightarrow_{\exists}$ to G_7 (that is not complete w.r.t. to $\rightarrow_{\sqsubseteq}$ rule) from the previous example we get …

General Inclusions (3)

Example



…this algorithm doesn't necessarily terminate ☹.

Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat "sufficiently frequently".

- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
  - **In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.**
- *exists*− rule is only applicable if the node a_1 in its definition is not blocked by another node.

Blocking in TA (2)

- In the previous example, the blocking ensures that node *“2″* is blocked by node *“0″* and no other expansion occurs. *Which model corresponds to such graph ?*
- **Introduced TA with subset blocking is sound, complete and finite decision procedure for $\mathcal{ALC}$.**

Let's play ...

- http://krizik.felk.cvut.cz/km/dl/index.html

9 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.
- Let's take a look, how to extend $\mathcal{ALC}$ while preserving decidability.

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

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

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 ?
- $Continent \equiv \{EUROPE, ASIA, AMERICA, AUSTRALIA, AFRICA, ANTARCTICA\}$?

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

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

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

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 ?
- 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

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

  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:
  - $\mathcal{SHOIN}$ 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:
  - 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 ∃-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.” ?
- ... we need rules, like

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

- in general, each variable can bind domain elements (similarly to undistinguished variables in the next lecture); however, such version is *undecidable*.

  DL-safe rules

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

Other extensions

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

Example 12. • (□ represents e.g. the ”believe” operator of an agent)

$$\Box(\mathit{Man} \sqsubseteq \mathit{Person} \sqcap \forall \mathit{hasFather} \cdot \mathit{Man}) \tag{1}$$

- 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:
- $$\Box(\mathit{Man} \Rightarrow \mathit{Person} \wedge \Box_{\mathit{hasFather}} \mathit{Man}) \tag{2}$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions

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

10 Conjunctive Queries

What we have ...

- Consistency checking is not enough. What if we would like to ask more, e.g. ... **How many czech writers died in the Czech Republic according to DBPedia ?** SELECT COUNT(?x) ?x ¡http://dbpedia.org/ontology/deathPlace¿ ¡http://dbpedia.org/resource/Czech_Republic¿ . ?x dcterms:subject ¡http://dbpedia.org/resource/Category:Czech_writers¿ at the following endpoint: `http://dbpedia-live.openlinksw.com/sparql/`

Query Types

Conjunctive (ABox) queries – queries asking for individual tuples complying with a graph-like pattern.

Metaqueries – queries asking for individual/concept/role tuples. There are several languages for metaqueries, e.g. SPARQL-DL, OWL-SAIQL, etc.

Example

In SPARQL-DL, the query "Find all people together with their type." can be written as follows :

$$Type(?x, ?c), SubClassOf(?c, Person)$$

Conjunctive (ABox) queries

Example

"Find all mothers and their daughters having at least one brother." :

$$\begin{aligned} Q(?x, ?z) \quad \leftarrow \quad & Woman(?x), hasChild(?x, ?y), hasChild(?x, ?z), \\ & Man(?y), Woman(?z) \end{aligned}$$

Conjunctive (ABox) queries are analogous to database SELECT-PROJECT-JOIN queries. A conjunctive query is in the form

$$Q(?x_1, \ldots, ?x_D) \leftarrow t_1, \ldots t_T,$$

where each t_i is either

- $C(y_k)$ (where C is a concept)
- $R(y_k, y_l)$ (where R is a role)

and y_i is either (i) an individual, or (ii) variable from a new set V (variables will be differentiated from individuals by the prefix "?"). Next, we need all $?x_i$ to be present also in one of t_i.

Conjunctive ABox Queries – Semantics

- Conjunctive queries of the form $Q()$ are called *boolean* – such queries only test existence of a relational structure in each model $\mathcal{I}$ of the ontology $\mathcal{K}$.
- Consider any interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. *Evaluation* η is a function from the set of individuals and variables into $\Delta^{\mathcal{I}}$ that coincides with $\mathcal{I}$ on individuals.
- Then $\mathcal{I} \models_{\eta} Q()$, iff
  - $\eta(y_k) \in C^{\mathcal{I}}$ for each atom $C(y_k)$ from $Q()$ and
  - $\langle \eta(y_k), \eta(y_l) \rangle \in R^{\mathcal{I}}$ for each atom $R(y_k, y_l)$ from $Q()$
- Interpretation $\mathcal{I}$ is a model of $Q()$, iff $\mathcal{I} \models_{\eta} Q()$ for some η.
- Next, $\mathcal{K} \models Q()$ ($Q()$ is satisfiable in $\mathcal{K}$) iff $\mathcal{I} \models Q()$ whenever $\mathcal{I} \models \mathcal{K}$

Conjunctive ABox Queries – Variables

- Queries without variables are not practically interesting. For queries with variables we define semantics as follows. An N-tuple $\langle i_1, \ldots, i_n \rangle$ is a *solution* to $Q(?x_1, \ldots, ?x_n)$ in theory $\mathcal{K}$, whenever $\mathcal{K} \models Q'()$, for a boolean query Q' obtained from Q by replacing all occurences of $?x_1$ in all t_k by an individual i_1, etc.
- In conjunctive queries two types of variables can be defined:

  distinguished occur in the query head as well as body, e.g. $?x, ?z$ in the previous example. These variables are evaluated as domain elements that are necessarily interpretations of some individual from $\mathcal{K}$. That individual is the binding to the distinguished variable in the query result.

  undistinguished occur only in the query body, e.g. $?y$ in the previous example. Their can be interpretated as any domain elements.

Conjunctive Queries – Examples

Example
Let's have a theory $\mathcal{K}_4 = (\emptyset, \{(\exists R_1 \cdot C_1)(i_1), R_2(i_1, i_2), C_2(i_2)\})$.

- Does $\mathcal{K} \models Q_1()$ hold for $Q_1() \leftarrow R_1(?x_1, ?x_2)$?
- What are the solutions of the query $Q_2(?x_1) \leftarrow R_1(?x_1, ?x_2)$ for $\mathcal{K}$?
- What are the solutions of the query $Q_3(?x_1, ?x_2) \leftarrow R_1(?x_1, ?x_2)$ for $\mathcal{K}$?

11 Evaluation of Conjunctive Queries in $\mathcal{ALC}$

Satisfiability of $\mathcal{ALC}$ Boolean Queries

- Satisfiability of the boolean query $Q()$ having a tree shape can be checked by means of the **rolling-up technique**.
  - Each two atoms $C_1(y_k)$ and $C_2(y_k)$ can be replaced by a single query atom of the form $(C_1 \sqcap C_2)(y_k)$.
  - Each query atom of the form $R(y_k, y_l)$ can be replaced by the term $(\exists R \cdot X)(y_k)$, if y_l occurs in at most one other query atom of the form $C(y_l)$ (if there is no $C(y_l)$ atom in the query, consider w.l.o.g. that C is $\top$). X equals to
    - (i) C, whenever y_l is a variable,
    - (ii) $C \sqcap Y_l$, whenever y_l is an individual. Y_l is a *representative concept* of individual y_l occuring neither in $\mathcal{K}$ nor in Q. For each y_l it is necessary to extend ABox of $\mathcal{K}$ with concept assertion $Y_l(y_l)$.

Satisfiability of $\mathcal{ALC}$ Boolean Queries (2)

…after rolling-up the query we obtain the query $Q()' \leftarrow C(y)$, that is satisfied in $\mathcal{K}$, iff $Q()$ is satisfied in $\mathcal{K}$:

- If y is an individual, then $Q'()$ is satisfied, whenever $\mathcal{K} \models C(y)$ (i.e. $\mathcal{K} \cup \{(\neg C)(y)\}$ is inconsistent)
- If y is a variable, then $Q'()$ is satisfied, whenever $\mathcal{K} \cup \{C \sqsubseteq \bot\}$ is inconsistent. Why ?

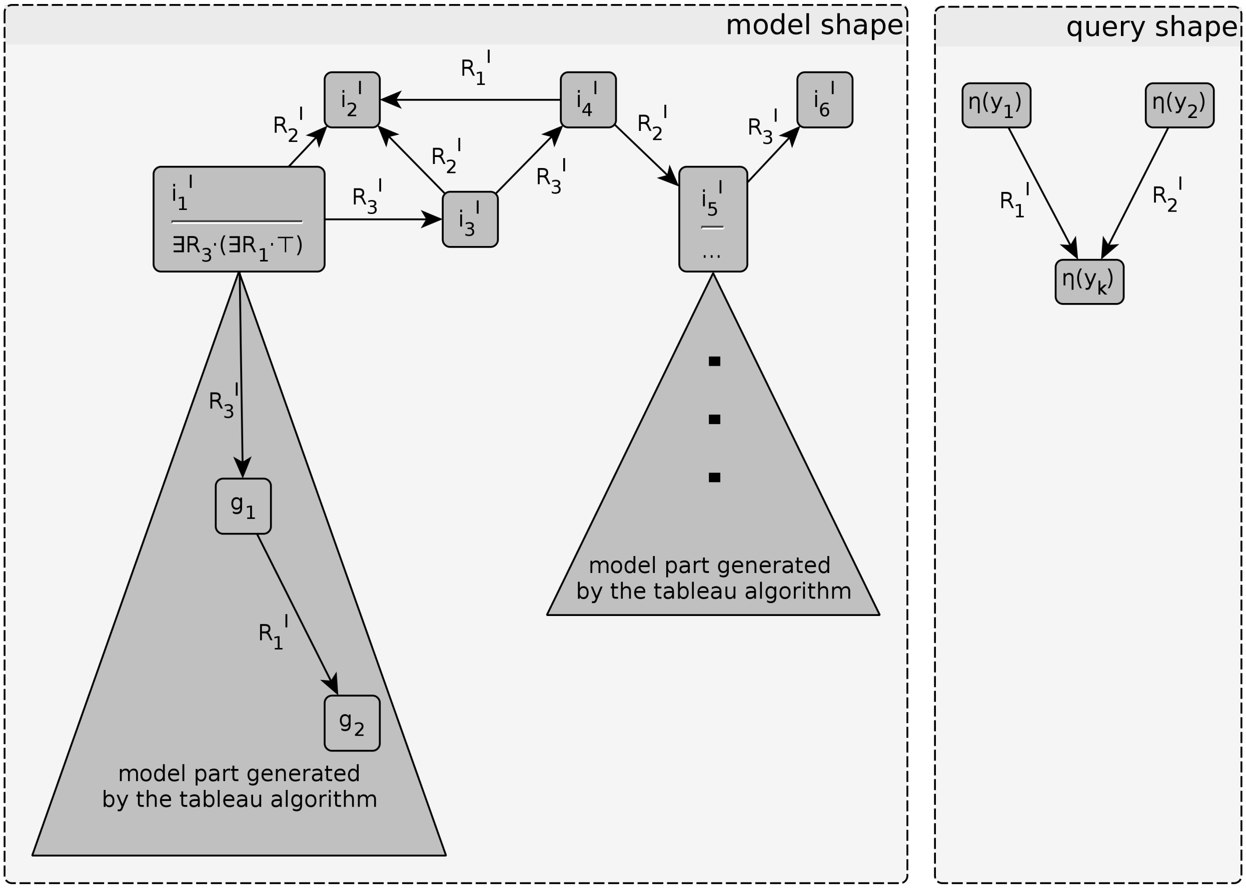
Example
Consider a query $Q_4() \leftarrow R_1(?x_1, ?x_2), R_2(?x_1, ?x_3), C_2(?x_3)$. This query can be rolled-up into the query $Q_4' \leftarrow (\exists R_1 \cdot \top \sqcap \exists R_2 \cdot C_2)(?x_1)$. This query is satisfiable in $\mathcal{K}_4$, as $\mathcal{K}_4 \cup \{(\exists R_1 \cdot \top \sqcap \exists R_2 \cdot C_2) \sqsubseteq \bot\}$ is inconsistent.

Satisfiability of Boolean Queries in $\mathcal{ALC}$ (3)

... and what to do with queries with distinguished variables ?

- Let's consider just queries that form "connected component" and contain for some variable y_k at least two query atoms of the form $R_1(y_1, y_k)$ and $R_2(y_2, y_k)$.
- Question: *Why is it enough to take just one connected component?*
- **Let's make use of the tree model property of $\mathcal{ALC}$. Each pair of atoms $R_1(y_1, y_k)$ and $R_2(y_2, y_k)$ can be satisfied only if y_k is interpreted as a domain element, that is an interpretation of an individual – y_k can be treated as distinguished. Why (see next slide) ?**
- For $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$ there is no sound and complete decision procedure for general boolean queries.

$\mathcal{ALC}$ Model Example



Queries with Distinguished Variables – naive pruning

Consider arbitrary query $Q(?x_1, \ldots, ?x_D)$. How to evaluate it ?

- **naive way**: Replace each distinguished variable x_i with each individual occuring in $\mathcal{K}$. *Solutions* are those D-tuples $\langle i_1, \ldots, i_D \rangle$, for which a boolean query created from Q by replacing each x_k with i_k is satisfiable.

Example 13. Remind that $\mathcal{K}_4 = (\emptyset, \{(\exists R_1 \cdot C_1)(i_1), R_2(i_1, i_2), C_2(i_2)\})$. The query

$$Q_5(?x_1) \leftarrow R_1(?x_1, ?x_2), R_2(?x_1, ?x_3), C_2(?x_3)$$

has *solution* $\langle i_1 \rangle$ as

$$Q'_5() \leftarrow R_1(i_1, ?x_2), R_2(i_1, ?x_3), C_2(?x_3)$$

can be rolled into $Q''_5()$ for which $\mathcal{K}_4 \models Q''_5$:

$$Q''_5() \leftarrow (\exists R_1 \cdot \top \sqcap \exists R_2 \cdot C_2)(i_1)$$

Queries with Distinguished Variables – naive pruning

... another example

Example 14. The query

$$Q_6(?x_1, ?x_3) \leftarrow R_1(?x_1, ?x_2), R_2(?x_1, ?x_3), C_2(?x_3)$$

has *solution* $\langle i_1, i_2 \rangle$ as

$$Q'_6() \leftarrow R_1(i_1, ?x_2), R_2(i_1, i_2), C_2(i_2)$$

can be rolled into Q''_6 for which $\mathcal{K}_4 \cup \{I_2(i_2)\} \models Q''_6$.

$$Q''_6() \leftarrow (\exists R_1 \cdot \top \sqcap \exists R_2 \cdot (C_2 \sqcap I_2))(i_1).$$

Similarly $Q_7(?x_1, ?x_2) \leftarrow R_1(?x_1, ?x_2), R_2(?x_1, ?x_3), C_2(?x_3)$ has no solution.

Queries with Distinguished Variables – iterative pruning

- **... a bit more clever strategy than replacing all variables**: First, let's replace just the first variable $?x_1$ with each individual from $\mathcal{K}$, resulting in Q_2. If the subquery of Q_2 containing all query atoms from Q_2 without distinguished variables is not a logical consequence of $\mathcal{K}$, then we do not need to test potential bindings for other variables.
- Many other optimizations are available.

Queries with Distinguished Variables – iterative pruning

Example 15. For the query $Q_6(?x_1, ?x_3)$, the naive strategy needs to check four different bindings (resulting in four tableau algorithm runs)

$$\langle i_1, i_1 \rangle,$$
$$\langle i_1, i_2 \rangle,$$
$$\langle i_2, i_1 \rangle,$$
$$\langle i_2, i_2 \rangle.$$

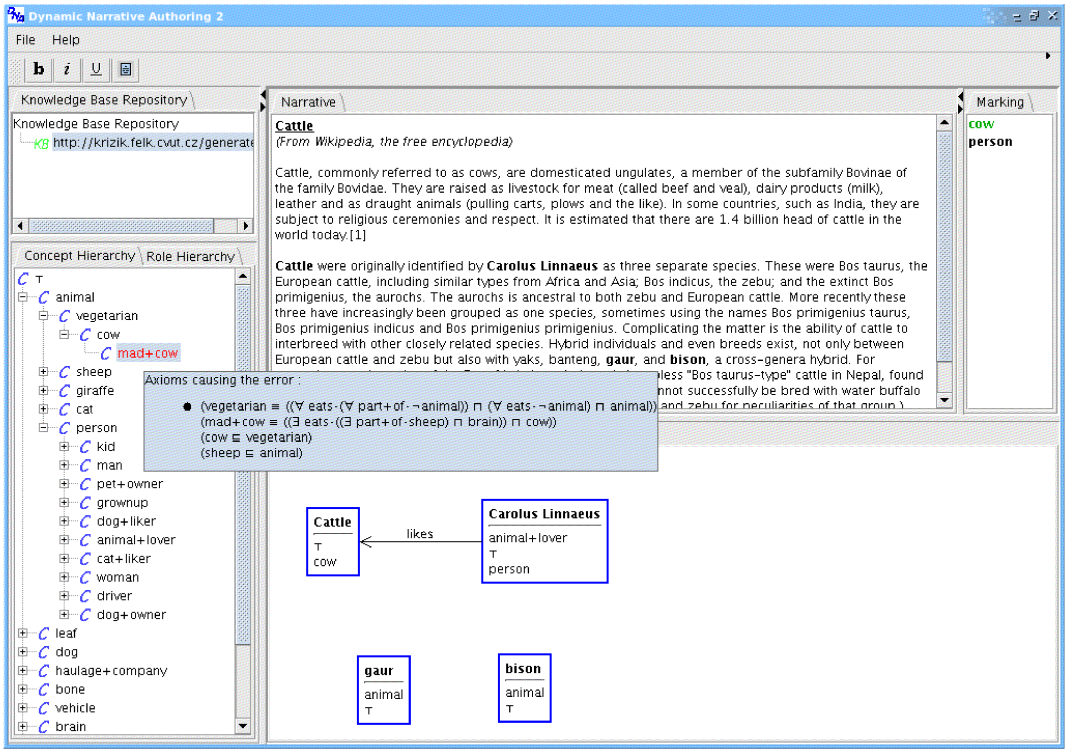
Out of them only $\langle i_1, i_2 \rangle$ is a solution for Q_6. Consider only partial binding $\langle i_2 \rangle$ for $?x_1$. Applying this binding to Q_6 we get $Q_7(?x_3) = R_1(i_2, ?x_2), R_2(i_2, ?x_3), C_2(?x_3)$. Its distinguished-variable-free subquery is $Q'_7() = R_1(i_2, ?x_2)$ and $\mathcal{K}_4 \not\models Q'_7$. Because of **monotonicity** of $\mathcal{ALC}$, we do not need to check the two bindings for $?x_3$ in this case which saves us one tableau algorithm run.

12 Modeling Error Explanation

Motivation

- When an inference engine claims inconsistency of an ($\mathcal{ALC}$) theory/unsatisfiability of an ($\mathcal{ALC}$) concept, **what can we do with it ?**
- We can start iterating through all axioms in the theory and look, "what went wrong".
- ... but hardly in case we have **hundred thousand axioms**
- A solution might be to ask the computer to *localize the axioms causing the problem for us.*

DNA



Cattle
(From Wikipedia, the free encyclopedia)

Cattle, commonly referred to as cows, are domesticated ungulates, a member of the subfamily Bovinae of the family Bovidae. They are raised as livestock for meat (called beef and veal), dairy products (milk), leather and as draught animals (pulling carts, plows and the like). In some countries, such as India, they are subject to religious ceremonies and respect. It is estimated that there are 1.4 billion head of cattle in the world today.[1]

Cattle were originally identified by **Carolus Linnaeus** as three separate species. These were Bos taurus, the European cattle, including similar types from Africa and Asia; Bos indicus, the zebu; and the extinct Bos primigenius, the aurochs. The aurochs is ancestral to both zebu and European cattle. More recently these three have increasingly been grouped as one species, sometimes using the names Bos primigenius taurus, Bos primigenius indicus and Bos primigenius primigenius. Complicating the matter is the ability of cattle to interbreed with other closely related species. Hybrid individuals and even breeds exist, not only between European cattle and zebu but also with yaks, banteng, **gaur**, and **bison**, a cross-genera hybrid. For

less "Bos taurus-type" cattle in Nepal, found

nnot successfully be bred with water buffalo

and zebu for peculiarities of that group.)

Axioms causing the error :

- (vegetarian ≡ ((∀ eats·(∀ part+of·¬animal)) ⊓ (∀ eats·¬animal) ⊓ animal))
  (mad+cow ≡ ((∃ eats·((∃ part+of·sheep) ⊓ brain)) ⊓ cow))
  (cow ⊑ vegetarian)
  (sheep ⊑ animal)

MUPS – example

Minimal unsatisfiability preserving subterminology (MUPS) is a minimal set of axioms responsible for concept unsatisfiability.

Example

Consider theory $\mathcal{K}_5 = (\{\alpha_1, \alpha_2, \alpha_3\}, \emptyset)$

$$\begin{aligned} \alpha_1 &: \quad Person \sqsubseteq \exists hasParent \cdot (Man \sqcap Woman) \sqcap \forall hasParent \cdot \neg Person, \\ \alpha_2 &: \quad Man \sqsubseteq \neg Woman, \\ \alpha_3 &: \quad Man \sqcup Woman \sqsubseteq Person. \end{aligned}$$

Unsatisfiability of *Person* comes independently from two axiom sets (MUPSes), namely $\{\alpha_1, \alpha_2\}$ and $\{\alpha_1, \alpha_3\}$. Check it yourself !

MUPS

Currently two approaches exist for searching all MUPSes for given concept:

black-box methods perform many satisfiability tests using existing inference engine.

- ☺ flexible and easily reusable for another (description) logic
- ☹ time consuming

glass-box methods all integrated into an existing reasoning (typically tableau) algorithm.

- ☺ efficient
- ☹ hardly reusable for another (description) logic.

Glass-box methods

- For $\mathcal{ALC}$ there exists a complete algorithm with the following idea:
  - tableau algorithm for $\mathcal{ALC}$ is extended in such way that it "remembers which axioms were used during completion graph construction".
  - for each completion graph containing a clash, the axioms that were used during its construction can be transformed into a MUPS.
- Unfortunately, complete glass-box methods do not exist for OWL-DL and OWL2-DL. The same idea (tracking axioms used during completion graph construction) can be used also for these logics, but only as a preprocessing reducing the set of axioms used by a black-box algorithm.

13 Black-box methods

Task formulation

- Let's have *a set of axioms* X of given DL and *reasoner* R for given DL. We want to find MUPSes for :
  1. concept unsatisfiability, '
  2. theory (ontology) inconsistency,
  3. arbitrary entailment.
- It can be shown (see [Kal06]) that w.l.o.g. we can deal only with *concept unsatisfiability*.
- **MUPS:** Let's denote $MUPS(C,Y)$ a minimal subset $MUPS(C,Y) \subseteq Y \subseteq X$ causing unsatisfiability of C.
- **Diagnose:** Let's denote $DIAG(C,Y)$ a minimal subset $DIAG(C,Y) \subseteq Y \subseteq X$, such that if $DIAG(C,Y)$ is removed from Y, the concept C becomes satisfiable.

Task formulation (2)

- Let's focus on concept C unsatisfiability. Denote

$$R(C,Y) = \left\{ \begin{array}{ll} true & \text{iff} Y \not\models (C \sqsubseteq \bot) \\ false & \text{iff} Y \models (C \sqsubseteq \bot)) \end{array} \right\}.$$

- There are many methods (see [dSW03]). We introduce just two of them:
  - Algorithms based on CS-trees.
  - Algorithm for computing a single MUPS[Kal06] + Reiter algorithm [Rei87].

13.1 Algorithms based on CS-trees

CS-trees

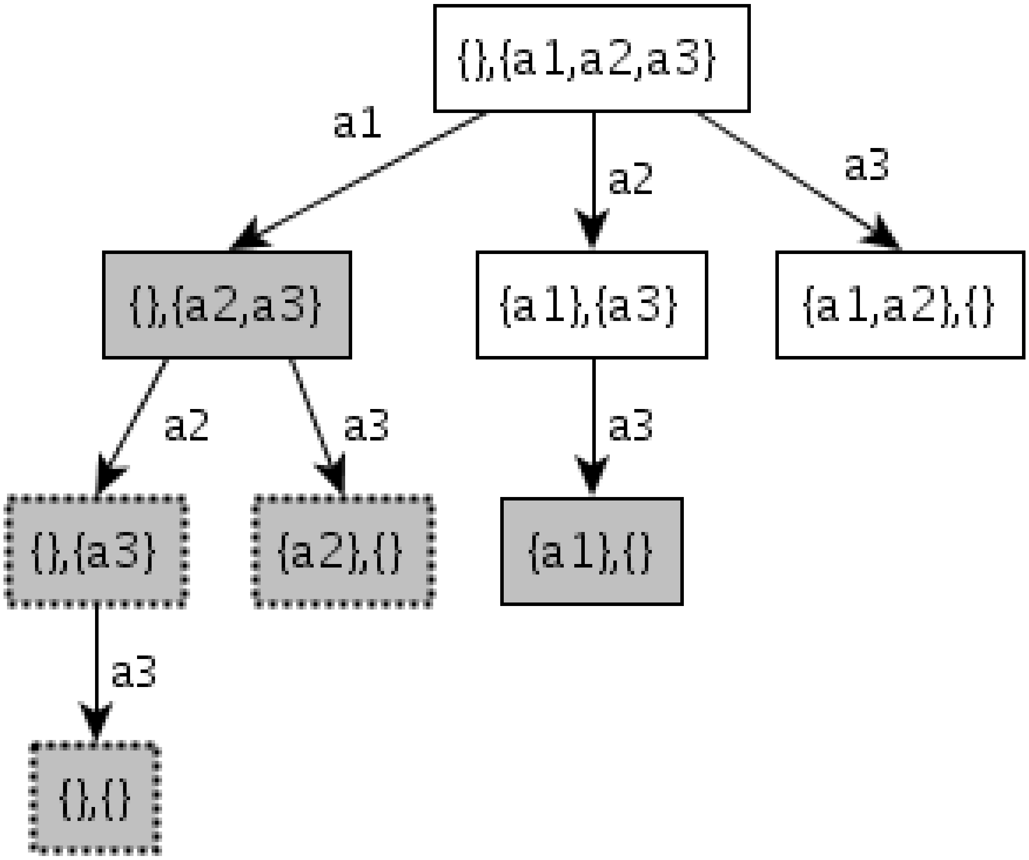
- A naive solution: test for each set of axioms from $\mathcal{T} \cup \mathcal{A}$ for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, whether the set causes unsatisfiability – minimal sets of this form are MUPSes.
- *Conflict-set trees (CS-trees)* systematize exploration of all these subsets of $\mathcal{T} \cup \mathcal{A}$. The main gist :

  If we found a set of axioms X that do not cause unsatisfiability of C (i.e. $X \nvDash C \sqsubseteq \bot$), then we know (and thus can avoid asking reasoner) that $Y \nvDash C \sqsubseteq \bot$ for each $Y \subseteq X$.
- CS-tree is a representation of the state space, where each state s has the form (D, P), where
  - D is a set of axioms that *necessarily has to be part of all MUPSes* found while exploring the subtree of s.
  - P is a set of axioms that *might be part of some MUPSes* found while exploring the subtree of s.

CS-tree Exploration – Example

Example
A CS-tree for unsatisfiability of *Person* (abbr. *Pe*, not to be mixed with the set P) in $\mathcal{K}_5 = \{\alpha_1, \alpha_2, \alpha_3\}$:

$$\underbrace{Pe \sqsubseteq \exists hP \cdot (M \sqcap W) \sqcap \forall hP \cdot \neg Pe}_{\alpha_1}, \quad \underbrace{M \sqsubseteq \neg W}_{\alpha_2}, \quad \underbrace{M \sqcup W \sqsubseteq Pe}_{\alpha_3}.$$



In gray states, the concept *Person* is satisfiable ($R(Pe, D \cup P) = true$). States with a dotted border are pruned by the algorithm.

CS-tree Exploration

The following algorithm is exponential in the number of tableau algorithm runs.

1 (Init) The root of the tree is an initial state $s_0 = (\emptyset, \mathcal{K})$ – apriori, we don't know any axiom being necessarily in a MUPS ($D_{s_0} = \emptyset$), but potentially all axioms can be there ($P_{s_0} = \mathcal{T} \cup \mathcal{A}$). Next, we define $Z = (s_0)$ and $R = \emptyset$

2 (Depth First Search) If Z is empty, stop the exploration. Otherwise pop the first element s from Z.

3 (Test) If $R(C, D_s \cup P_s) = true$ then no subset of $D_s \cup P_s$ can cause unsatisfiability – we continue with step 2.

4 (Finding an unsatisfiable set) We add $D_s \cup P_s$ into R and remove from R all $s' \in R$ such that $D_s \cup P_s \subseteq s'$. For $P_s = \alpha_1, \dots, \alpha_N$ we push to Z a new state $(D_s \cup \{\alpha_1, \dots, \alpha_{i-1}\}, P_s \setminus \{\alpha_1, \dots, \alpha_i\})$ – we continue with step 2.

CS-tree Exploration (2)

- Soundness : Step 4 is important – here, we cover all possibilities. It always holds that $D_s \cup P_s$ differs to $D'_s \cup P'_s$ by just one element, where s' is a successor of s.
- Finiteness : Set $D_s \cup P_s$ is finite at the beginning and gets smaller with the tree depth. Furthermore, in step 4 we generate only finite number of states.

13.2 Algorithm based on Reiter's Algorithm

Another Approach – Reiter's Algorithm

There is an alternative to CS-trees:

1. Find a single (arbitrary) MUPS ($singleMUPS$ in the next slides).
2. "remove the source of unsatisfiability provided by MUPS" (Reiter's algorithm in the next slides) from the set of axioms go explore the remaining axioms in the same manner.

13.3 Algorithm based on Reiter's Algorithm

Finding a single $MUPS(C, Y)$ – example

Example

The run of $singleMUPS(Person, \mathcal{K}_5)$ introduced next.

1.PHASE :

$\mathcal{K}_5 = \{\alpha_1, \alpha_2, \alpha_3\}$ $\quad R(Person, \{\alpha_1, \alpha_2\}) = false$
$S = \{\alpha_1, \alpha_2\}$

1.PHASE :

$\mathcal{K}_5 = \{\alpha_1, \alpha_2, \alpha_3\}$ $\quad R(Person, \{\alpha_1, \alpha_2\}) = false$
$S = \{\alpha_1, \alpha_2\}$

2.PHASE :

$S = \{\alpha_1, \alpha_2\}$ $\quad R(Person, \{\alpha_1, \alpha_2\} - \{\alpha_2\}) = true$
$K = \{\alpha_1, \alpha_2\}$

1.PHASE :

$\mathcal{K}_5 = \{\alpha_1, \alpha_2, \alpha_3\}$ $\quad R(Person, \{\alpha_1\}) = true$
$S = \{\alpha_1\}$

1.PHASE :

$\mathcal{K}_5 = \{\alpha_1, \alpha_2, \alpha_3\}$ $\quad R(Person, \{\alpha_1, \alpha_2\}) = false$
$S = \{\alpha_1, \alpha_2\}$

2.PHASE :

$S = \{\alpha_1, \alpha_2\}$ $\quad R(Person, \{\alpha_1, \alpha_2\} - \{\alpha_1\}) = true$
$K = \{\alpha_1\}$

singleMUPS(C, Y) – **finding a single MUPS**

The following algorithm is polynomial in the number of tableau algorithm applications – the computational complexity stems from the complexity of tableau algorithm itself.

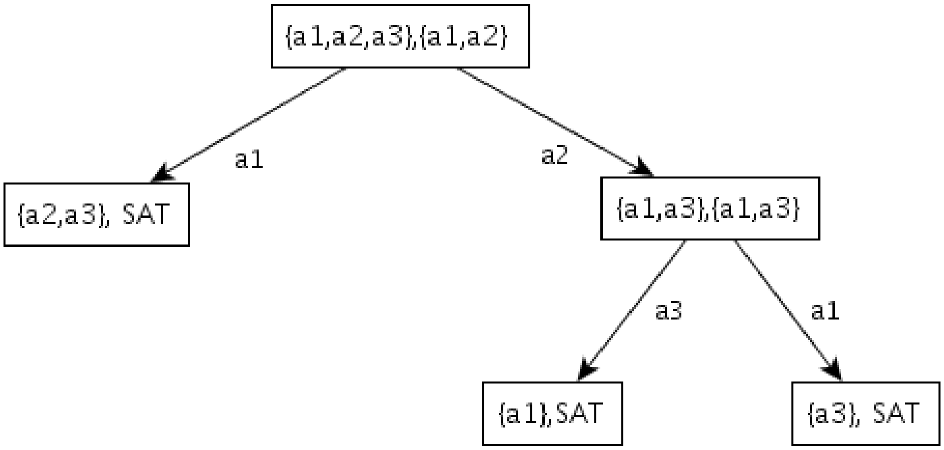
1 (Initialization) Denote $S = \emptyset$, $K = \emptyset$

2 (Finding superset of MUPS) While $R(C, S) = false$, then $S = S \cup \{\alpha\}$ for some $\alpha \in Y \setminus S$.

3 (Pruning found set) For each $\alpha \in S \backslash K$ evaluate $R(C, S \backslash \{\alpha\})$. If the result is $false$, then $K = K \cup \{\alpha\}$. The resulting K is itself a MUPS.

Finding all MUPSes – Reiter Algorithm, example

Example (continued)



The algorithm ends up with two MUPSes $\{\alpha_1, \alpha_2\}$ a $\{\alpha_1, \alpha_3\}$. "For free" we got diagnoses $\{\alpha_1\}$ a $\{\alpha_2, \alpha_3\}$.

Finding all MUPSes – Reiter Algorithm

- Reiter algorithm runs *singleMUPS*(C, Y) multiple times to construct so called "Hitting Set Tree", nodes of which are pairs $(\mathcal{K}_i, M_i)$, where $\mathcal{K}_i$ lacks some axioms comparing to $\mathcal{K}$ and $M_i = singleMUPS(C, \mathcal{K}_i)$, or $M_i = \text{“}SAT''$, if C is satisfiable w.r.t. $\mathcal{K}_i$.
- Paths from the root to leaves build up *diagnoses* (i.e. minimal sets of axioms, each of which removed from $\mathcal{K}$ causes satisfiability of C).
- Number of *singleMUPS*(C, Y) calls is at most exponential w.r.t. the initial axioms count. Why ?

Finding all MUPSes – Reiter Algorithm (2)

1 (Initialization) Find a single MUPS for C in $\mathcal{K}$, and construct the root $s_0 = (\mathcal{K}, singleMUPS(C, \mathcal{K}))$ of the hitting set tree. Next, set $Z = (s_0)$.

2 (Depth First Search) If Z is empty, STOP.

3 (Test) Otherwise pop an element from Z and denote it as $s_i = (\mathcal{K}_i, M_i)$. If $M_i = \text{“}SAT''$, then go to step 2.

4 (Decomposition) For each $\alpha \in M_i$ insert into Z a new node $(\mathcal{K}_i \setminus \{\alpha\}, singleMUPS(\mathcal{K}_i \setminus \{\alpha\}, C))$. Go to step 2.

Modeling Error Explanation – Summary

- finding MUPSes is the most common way for explaining modeling errors.
- black-box vs. glass box methods. Other methods involve e.g. incremental methods [dSW03].
- the goal is to find MUPSes (and diagnoses) – what to do in order to solve a modeling problem (unsatisfiability,inconsistency).
- above mentioned methods are quite universal – they can be used for many other problems that are not related with description logics.

References

References

[MvL13] * Vladimír Mařík, Olga Štěpánková, and Jiří Lažanský. *Umělá inteligence 6 [in czech], Chapters 2-4*. Academia, 2013.

[BCM$^+$03] * 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.

[BCM$^+$03] * Enrico Franconi. *Course on Description Logics*. http://www.inf.unibz.it/ franconi/dl/course/, cit. 22.9.2013.

[Sow00] John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole, 2000.

[MvL93] Vladimír Mařík, Olga Štěpánková, and Jiří Lažanský. *Umělá inteligence 1*. Academia, 1993.

[Rei87] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–96, April 1987.

[dSW03] Maria Garcia de la Banda, Peter J. Stuckey, and Jeremy Wazny. Finding All Minimal Unsatisfiable Subsets. In *Proceedings of PPDP'03*, 2003.

[Kal06] Aditya Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, University of Maryland, 2006.