

Description Logics & Ontology Languages — an Introduction and Overview

Uli Sattler

uli.sattler@manchester.ac.uk

Department of Computer Science

University of Manchester

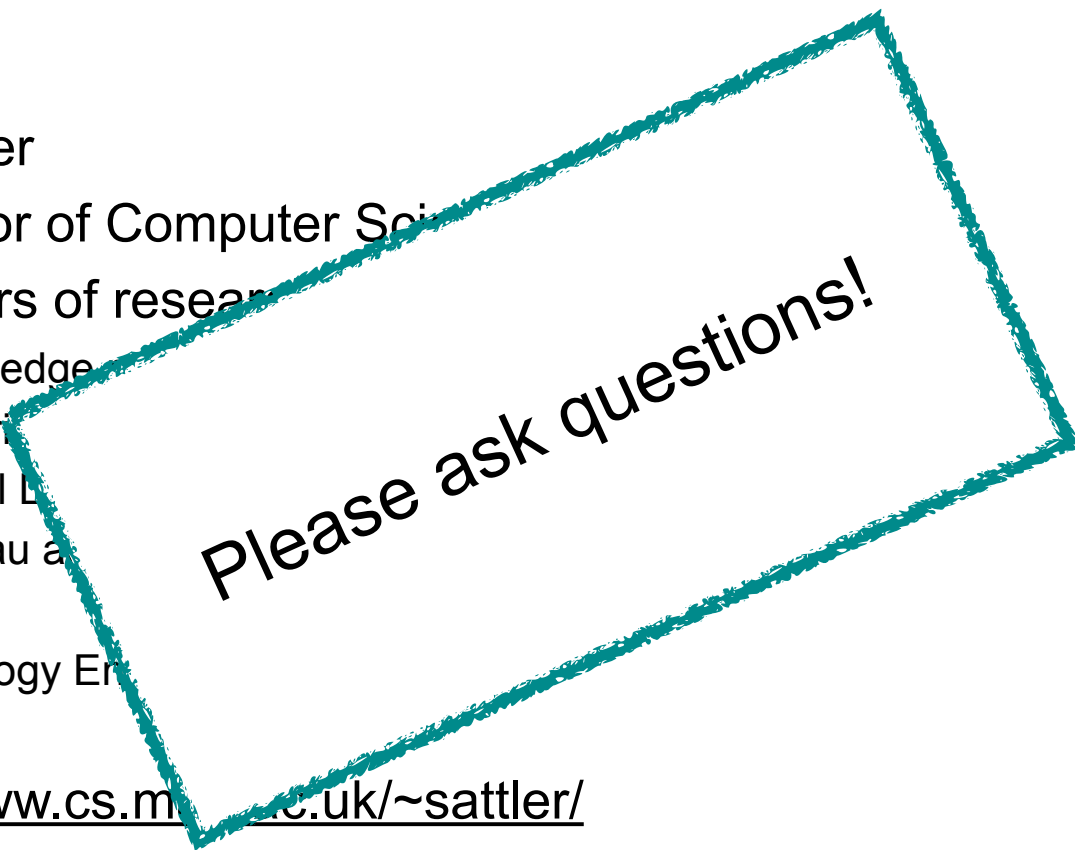
TEASE-LP 2020

About me

- Uli Sattler
- Professor of Computer Science in Manchester
- 25+ years of research in
 - knowledge representation and reasoning
 - Description Logics
 - Modal Logic
 - tableau algorithms
 - OWL
 - Ontology Engineering
 - ...
- <http://www.cs.man.ac.uk/~sattler/>

About me

- Uli Sattler
- Professor of Computer Science
- 25+ years of research
 - knowledge
 - Description
 - Modal Logic
 - tableau algorithms
 - OWL
 - Ontology Engineering
 - ...
- <http://www.cs.manchester.ac.uk/~sattler/>



Please ask questions!

Outline for today

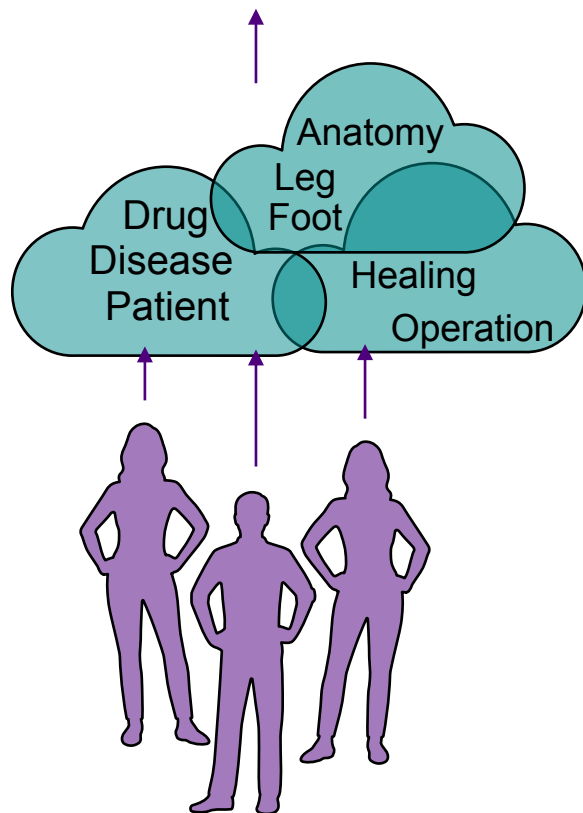
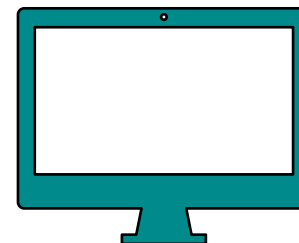
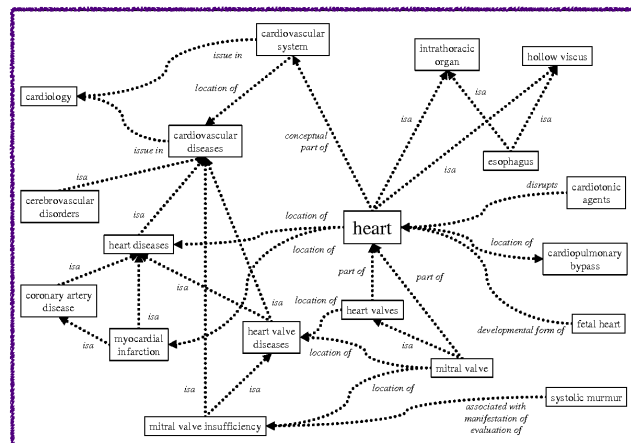
- Setting the scene:
 - how and why people tame logics
- A brief history of Description Logics (DLs)
- Meet \mathcal{ALC} , a DL
 - syntax & semantics
 - relationship to FOL & Modal Logic
 - core reasoning problems/services
 - tableau (chase) for \mathcal{ALC}
- Dessert:
 - explanation of entailments/inferences
 - modularity in DLs

Why/how we tame logics?

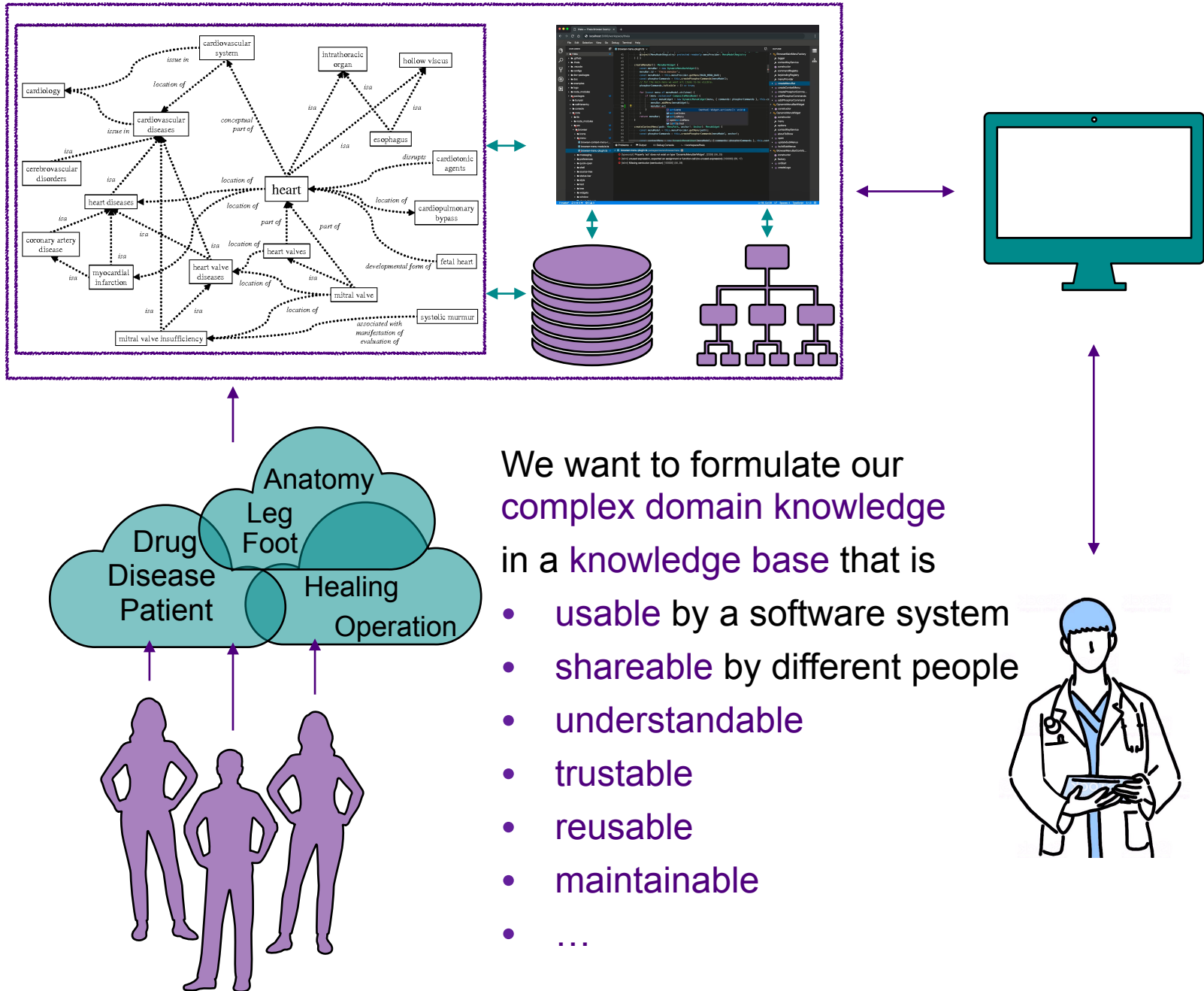
Logics for Knowledge Representation

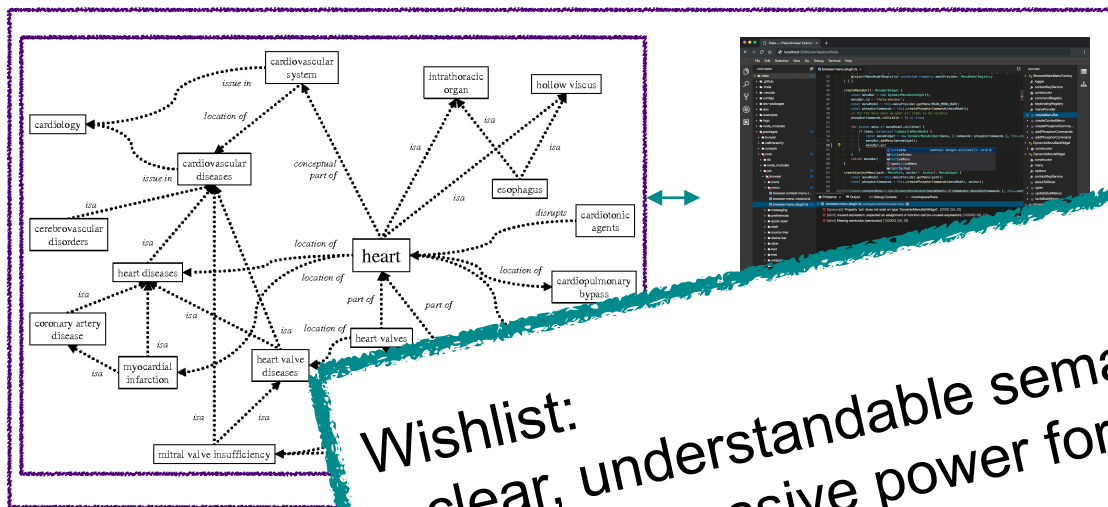
We want to make our **complex domain knowledge**

- e.g., about medicine,
- **usable** by a software system
- **shareable** by different people

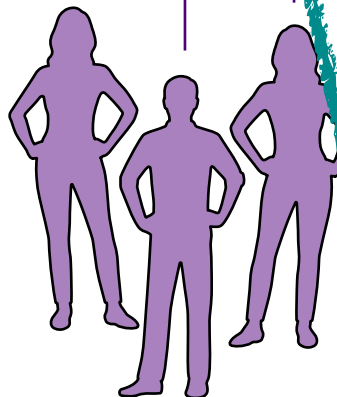
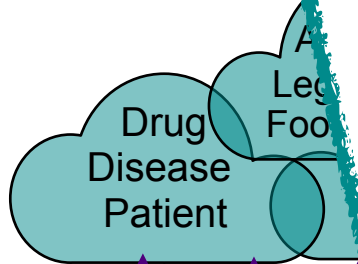




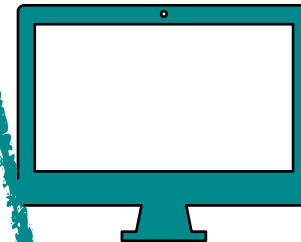




- Wishlist:**
- clear, understandable semantics
 - good expressive power for
 - domain experts
 - tools
 - needs to work for SEs
 - stable syntax
 - annotations
 - ...

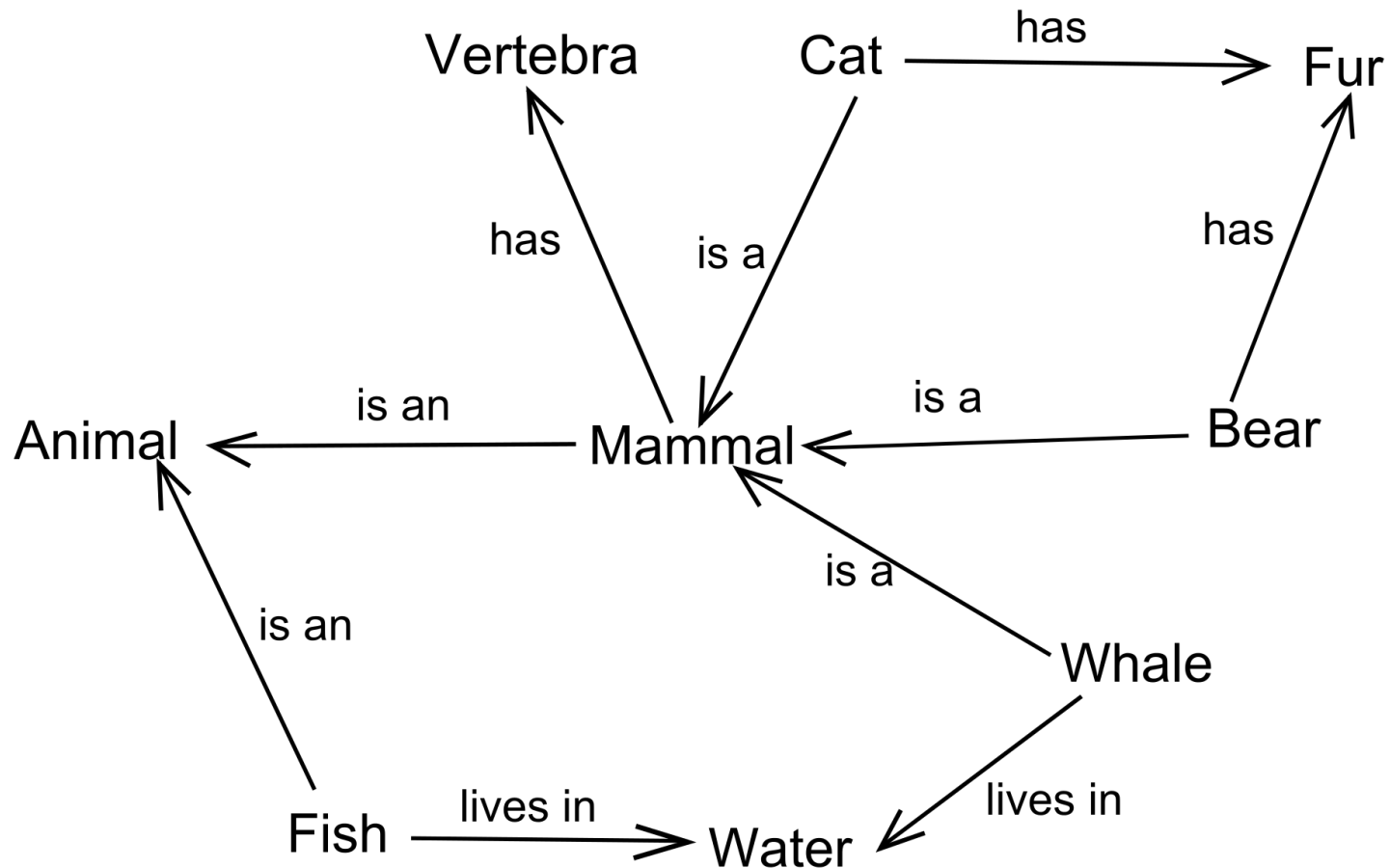


- trustable
- reusable
- maintainable
- ...



How it all began...

- Semantic Networks

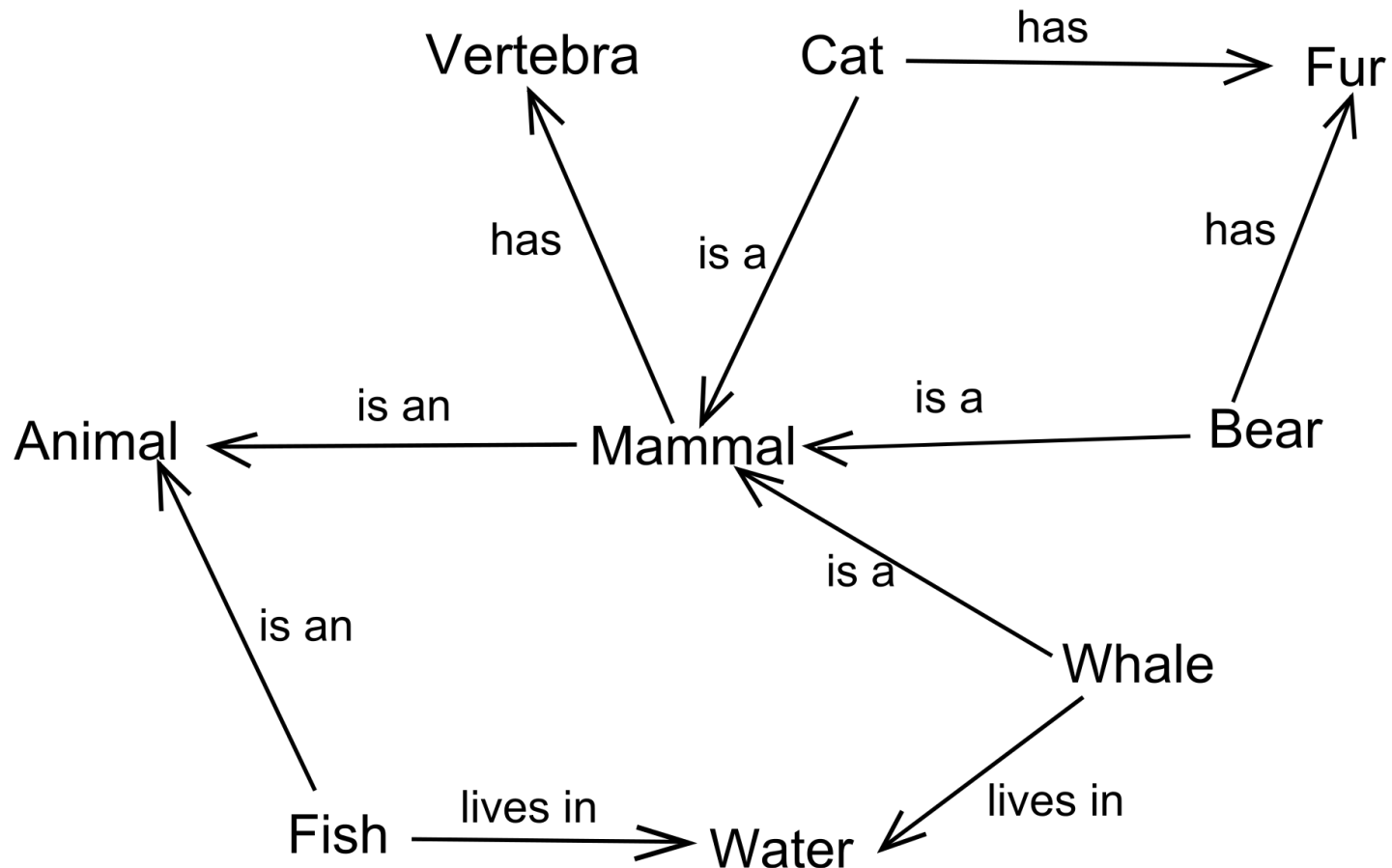


How it all began...

- Semantic Networks

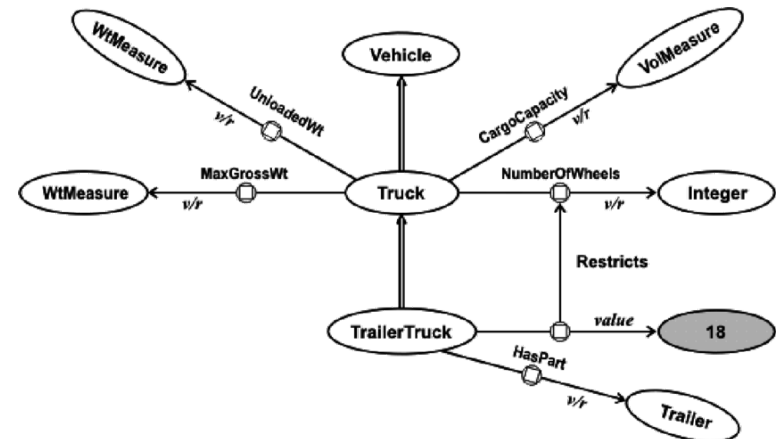
Questions:

- is every Whale an Animal?
- can Cats have other things than Fur?
- do Cats have a Vertebra?
-



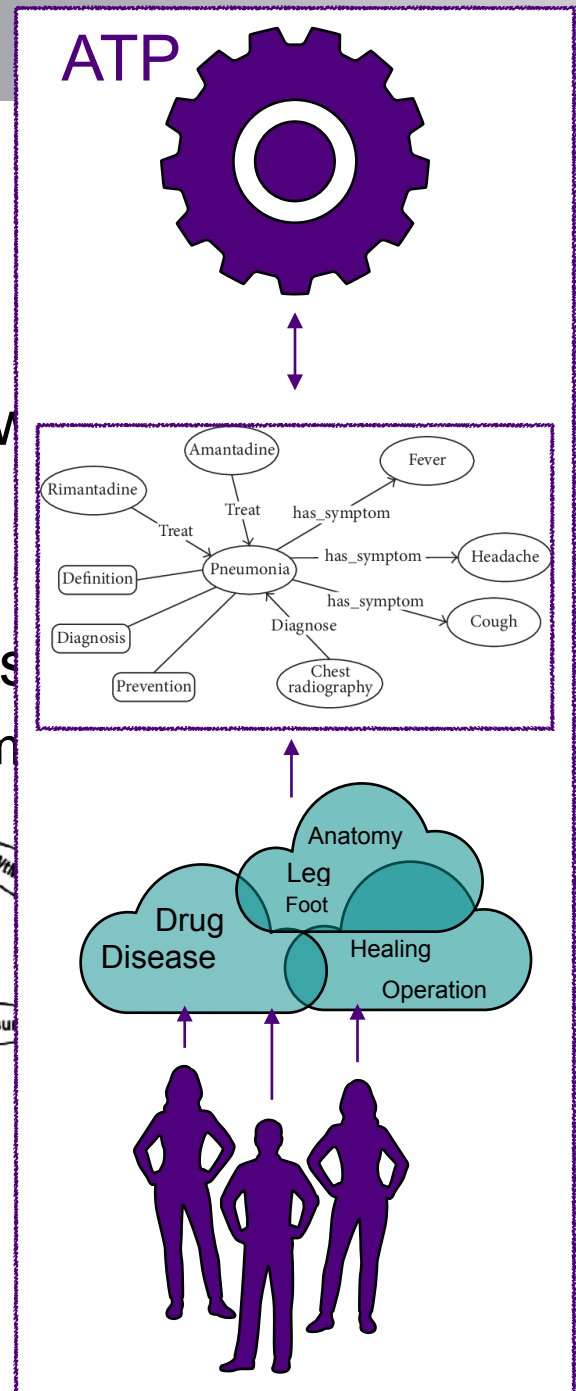
Logic to the rescue!

- Describe meaning of pictures/networks by translating them to **logic**
 - gives well-defined semantics!
- mid-80s Brachmann & Schmolze's **KL-ONE**
 - a powerful, logic-based KR formalism
 - with tools, reasoners, ...
- 1989 Schmidt-Schauß:
“*KL-ONE is undecidable!*”
 - **not** good expressive power for tools
 - too high



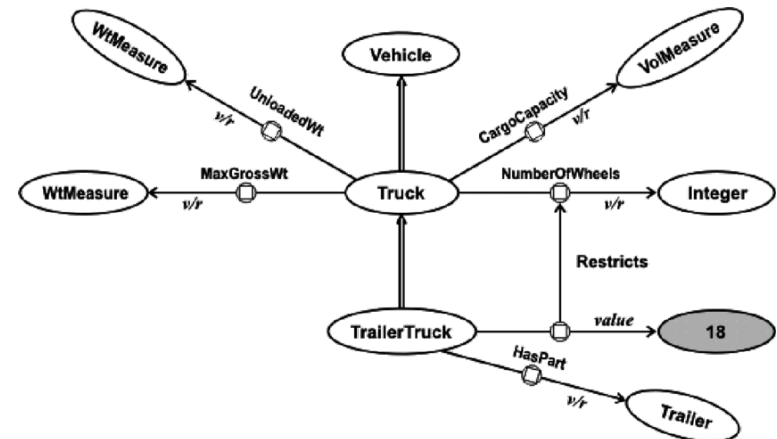
Logic to the rescue!

- Describe meaning of pictures/networks by translating them to **logic**
 - gives well-defined semantics!
- mid-80s Brachmann & Schmolze's
 - a powerful, logic-based KR formalism
 - with tools, reasoners, ...
- 1989 Schmidt-Schauß:
 - "KL-ONE is undecidable!"*
 - not** good expressive power for tools
 - too high



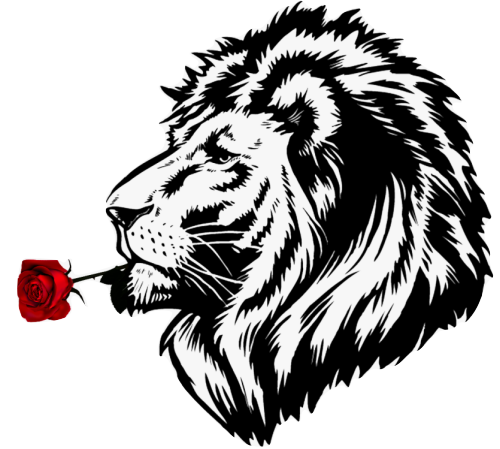
Logic to the rescue!

- Describe meaning of pictures/networks by translating them to **logic**
 - gives well-defined semantics!
- mid-80s Brachmann & Schmolze's **KL-ONE**
 - a powerful, logic-based KR formalism
 - with tools, reasoners, ...
- 1989 Schmidt-Schauß:
 - "KL-ONE is undecidable!"*
 - not** good expressive power for tools
 - too high
- We need to *tame* underlying logic



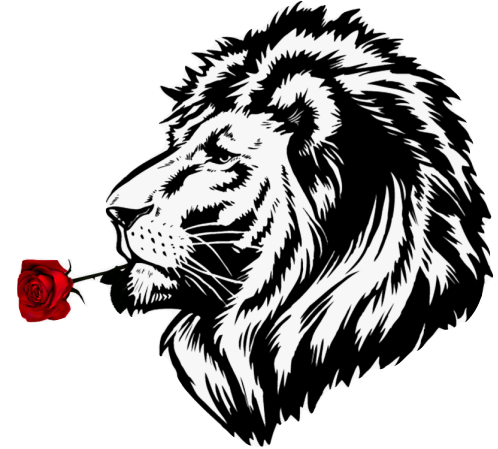
How to tame logic

- Reduce non-determinism
 - Horn fragments/clauses
- Restrict size of universe
 - *no existentials in the head*
 - Logic Programming
- Restrict number of variables
 - L2, C2
- Restrict quantifier alternation
 - Bernays–Schönfinkel class
- Restrict shape of universe/localise quantifiers
 - Modal Logic
 - Description Logic
 - Guarded Fragment



How to tame logic

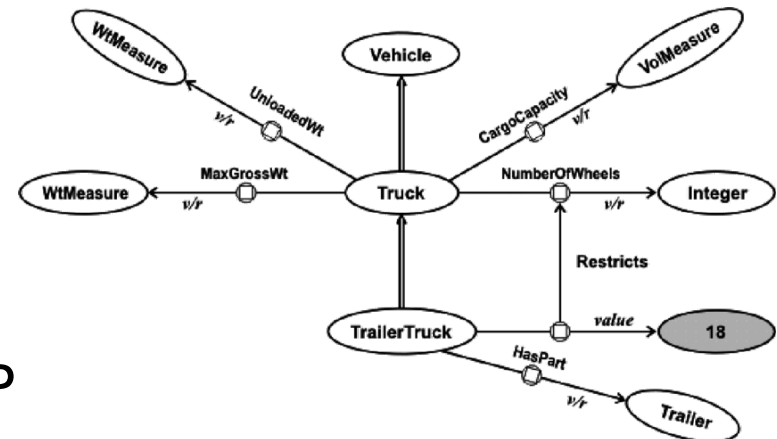
- Reduce non-determinism
 - Horn fragments/clauses
- Restrict size of universe
 - *no existentials in the head*
 - Logic Programming
- Restrict number of variables
 - L2, C2
- Restrict quantifier alternation
 - Bernays–Schönfinkel class
- Restrict shape of universe/localise quantifiers
 - Modal Logic
 - Description Logic
 - Guarded Fragment



Description Logics - an overview

Description Logics

- are designed for **Knowledge Representation**
- started in mid-80s with Brachmann & Schmolze's **KL-ONE**
 - a powerful, logic-based KR formalism
 - with tools, reasoners, ...
 - terminological knowledge representation systems*
 - concept languages*
- now a relevant part of KR&R
 - logic foundations of
 - Semantic Web
 - ontology languages
- have common off-springs with LP
 - Datalog+/-,
 - Datalog with (limited) existentials in the head



Description Logics

- are designed for **Knowledge Representation**
- **ontology** (a DL Knowledge Base) consists of
 - ⇒ **TBox** for terminological knowledge

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$

Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$

hasLoc o isPartOf \sqsubseteq hasLoc

Description Logics

- are designed for **Knowledge Representation**
- **ontology** (a DL Knowledge Base) consists of
 - ⇒ **TBox** for terminological knowledge

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$
 Inflammation \sqsubseteq Disease
 HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$
 Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$
 Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$
 hasLoc o isPartOf \sqsubseteq hasLoc

- ⇒ **ABox** for assertions or facts

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation } \sqcap \exists \text{hasLoc.Endocardium)})$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into **FOL**

TBox

DL: Inflammation \sqsubseteq Disease

FOL: $\forall x. \text{Inflammation}(x) \Rightarrow \text{Disease}(x)$

DL: HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

FOL: $\forall x. \text{HeartDisease}(x) \Leftrightarrow (\text{Disease}(x) \wedge \exists y. (\text{hasLoc}(x, y) \wedge \text{Heart}(y)))$

DL: hasLoc o isPartOf \sqsubseteq hasLoc

FOL: $\forall x, y, z. \text{hasLoc}(x, y) \wedge \text{isPartOf}(y, z) \Rightarrow \text{hasLoc}(x, z)$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into **FOL**

TBox

DL: Inflammation \sqsubseteq Disease

FOL: $\forall x. \text{Inflammation}(x) \Rightarrow \text{Disease}(x)$

DL: HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

FOL: $\forall x. \text{HeartDisease}(x) \Leftrightarrow (\text{Disease}(x) \wedge \exists y. (\text{hasLoc}(x, y) \wedge \text{Heart}(y)))$

*Existential
in head!*

DL: hasLoc o isPartOf \sqsubseteq hasLoc

FOL: $\forall x, y, z. \text{hasLoc}(x, y) \wedge \text{isPartOf}(y, z) \Rightarrow \text{hasLoc}(x, z)$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into **FOL**

ABox:

DL: Bob:(Person \sqcap \exists suffersFrom.Endocarditis)

FOL: Person(Bob) \wedge $\exists y$.suffersFrom(Bob, y) \wedge Endocarditis(y)

DL: (Bob,Mary):hasMother

FOL: hasMother(Bob, Mary)

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- are also closely related to **modal logics**
 - thus to 2-variable/guarded fragment

DL: Endocarditis \sqsubseteq Inflammation $\sqcap \exists \text{hasLoc. Endocardium}$

ML: $[u](\neg \text{Endocarditis} \vee (\text{Inflammation} \wedge \langle \text{hasLoc} \rangle \text{Heart}))$

DL: HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc. Heart}$

ML: $[u](\neg \text{HeartDisease} \vee (\text{Disease} \wedge \langle \text{hasLoc} \rangle \text{Heart})$
 $\text{HeartDisease} \vee \neg (\text{Disease} \wedge \langle \text{hasLoc} \rangle \text{Heart}))$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- with its own terminology

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$
Inflammation \sqsubseteq Disease
HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$
Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$
Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$
hasLoc o isPartOf \sqsubseteq hasLoc

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation } \sqcap \exists \text{hasLoc.Endocardium)})$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- with its own terminology

unary pred.:
Concept

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$

Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$

hasLoc o isPartOf \sqsubseteq hasLoc

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation } \sqcap \exists \text{hasLoc.Endocardium))}$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- with its own terminology

binary pred.:
Role

unary pred.:
Concept

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$

Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$

hasLoc o isPartOf \sqsubseteq hasLoc

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation } \sqcap \exists \text{hasLoc.Endocardium))}$

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- with its own terminology

binary pred.:
Role

unary pred.:
Concept

Patient \equiv Person $\sqcap \exists$ suffersFrom.Disease
 Inflammation \sqsubseteq Disease
 HeartDisease \equiv Disease $\sqcap \exists$ hasLoc.Heart
 Endocarditis \equiv Inflammation $\sqcap \exists$ hasLoc.Endocardium
 Endocardium \sqsubseteq Bodypart $\sqcap \exists$ isPartOf.Heart
 isPartOf \sqsubseteq hasLoc

constant:
Individual

Bob:(Person \sqcap
 \exists suffersFrom.(Inflammation $\sqcap \exists$ hasLoc.Endocardium))

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- come with classical FOL *semantics*

Patient \equiv Person $\sqcap \exists \text{ suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{ hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{ hasLoc.Endocardium}$

Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{ isPartOf.Heart}$

hasLoc o isPartOf \sqsubseteq hasLoc

\models Endocarditis \sqsubseteq HeartDisease

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- come with classical FOL *semantics*

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$

Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$

hasLoc o isPartOf \sqsubseteq hasLoc

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation} \sqcap \exists \text{hasLoc.Endocardium})$)

\models Bob:Patient

Description Logics

- sit nicely/mostly between Propositional and FO-Logic
- can be translated into FOL
- come with classical FOL *semantics*

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$

Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$

hasLoc o isPartOf \sqsubseteq hasLoc

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation } \sqcap \exists \text{hasLoc.Endocardium))}$

\models Bob: Patient $\sqcap \exists \text{suffersFrom.HeartDisease}$

Description Logics Reasoning Tasks

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$ for each $\alpha \in O$

Description Logics Reasoning Tasks

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$ for each $\alpha \in O$

Consistency

Description Logics Reasoning Tasks

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$ for each $\alpha \in O$

Consistency

- a concept A , test whether $O \models A \sqsubseteq \perp$

Satisfiability

Description Logics Reasoning Tasks

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$ for each $\alpha \in O$

Consistency

- a concept A , test whether $O \models A \sqsubseteq \perp$

Satisfiability

- two concepts A, B , test whether $O \models A \sqsubseteq B$

Subsumption

Description Logics Reasoning Tasks

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$ for each $\alpha \in O$

Consistency

- a concept A , test whether $O \models A \sqsubseteq \perp$

Satisfiability

- two concepts A, B , test whether $O \models A \sqsubseteq B$

Subsumption

- individual b , concept name A , test whether $O \models b:A$

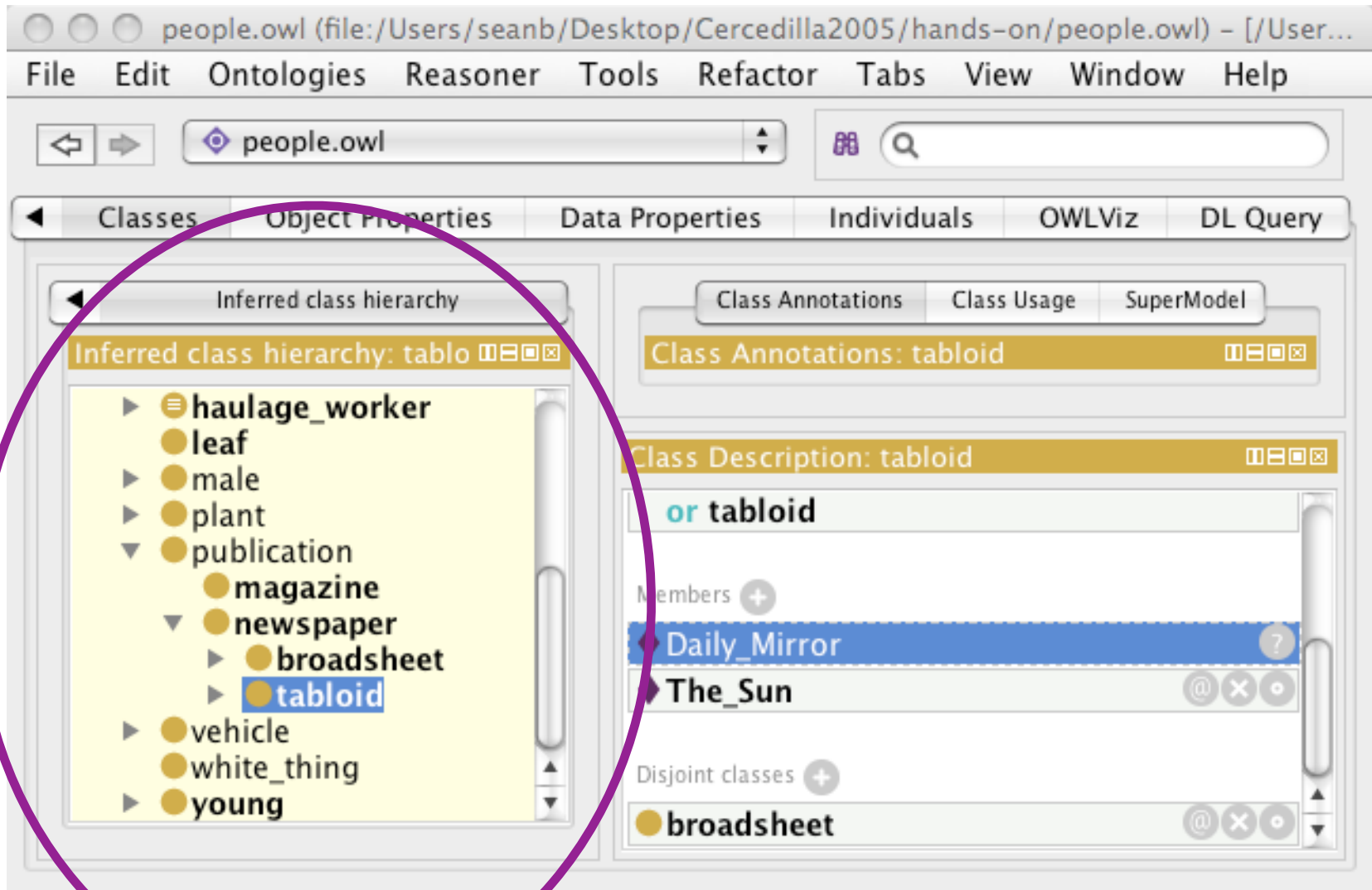
Instancship

Description Logics Reasoning Tasks

Classification given ontology O , test

- consistency of O
 - for each concept name A , whether $O \models A \sqsubseteq \perp$
 - for each concept names A, B , whether $O \models A \sqsubseteq B$
 - for each individual b , concept name A , whether $O \models b:A$
- ⇒ *inferred concept hierarchy*

Description Logics Reasoning Tasks



Description Logics Reasoning Tasks

Classification given ontology O , test

- consistency of O
 - for each concept name A , whether $O \models A \sqsubseteq \perp$
 - for each concept names A, B , whether $O \models A \sqsubseteq B$
 - for each individual b , concept name A , whether $O \models b:A$
- ⇒ *inferred concept hierarchy*

n^2 entailment
tests

Description Logics Reasoning Tasks

Classification given ontology O , test

n^2 entailment tests

- consistency of O
- for each concept name A , whether $O \models A \sqsubseteq \perp$
- for each concept names A, B , whether $O \models A \sqsubseteq B$
- for each individual b , concept name A , whether $O \models b:A$

⇒ *inferred concept hierarchy*

- *complexity* of each entailment test $O \models? \dots$ is
 - between AC_0
 - via polynomial
 - to NExpTime-complete



Complexity of Entailment Checking

| Name | Meaning |
|-------------------|---------------------------|
| First Order Logic | Undecidable |
| ... | |
| NExpTime | non-det. exponential time |
| ExpTime | exponential time |
| PSpace | polynomial space |
| NP | non-det. polynomial time |
| P | polynomial time |
| L | logarithmic space |

Complexity of Entailment Checking

| Name | Meaning | |
|-------------------|-------------|--|
| First Order Logic | Undecidable | no sound, complete & terminating algorithm |
| | ... | KL-One |
| | | ALC(D) |
| | | |
| | NExpTime | non-det. exponential time |
| | | SROIQ |
| | | SHIQ |
| | ExpTime | exponential time |
| | | ALCIQ |
| | | ... |
| | | ALC |
| | | ALCI |
| | | ALCt |
| | PSpace | polynomial space |
| Boolean Logic | NP | non-det. polynomial time |
| | | |
| | P | polynomial time |
| | | EL |
| | L | logarithmic space |
| | | DL-lite |

Description Logics Reasoning Tasks

- *classification*: given ontology O , test

n^2 entailment tests

- consistency of O
- for each concept name A , whether $O \models A \sqsubseteq \perp$
- for each concept names A, B , whether $O \models A \sqsubseteq B$
- for each individual b , concept name A , whether $O \models b:A$

⇒ *inferred concept hierarchy*

- *complexity* of each entailment test $O \models? \dots$ is

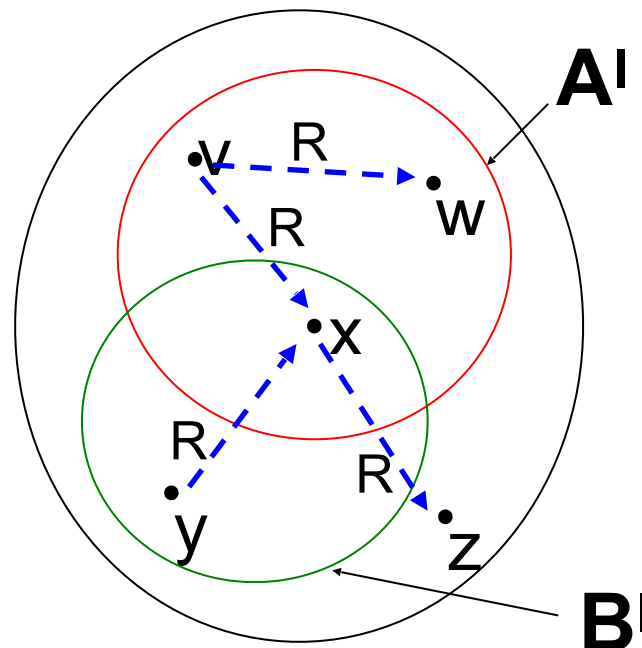
- between AC_0
- via polynomial
- to NExpTime-complete



- specialised *DL reasoners* are implemented & optimised
 - ORE reasoner competitions:
 - > 10 top notch reasoners
 - able to classify large, complex ontologies with >300K terms

Description Logic Models

- as FOL fragments, they have structured models



Description Logic Models

- as FOL fragments, they have **structured models**:
 - most DLs have a **tree model property**
 - despite constants, transitive roles
 - some have the **finite model property**:
 - *ALC*: $\Box, \sqcup, \neg, \exists, \forall$
 - others lack it: *ALC* extended with
 - inverse/converse roles and
e.g., both *hasLoc* and *isLocatedIn*
 - number restrictions can enforce **infinite models**
e.g., $(\leq 1 \text{ hasMother})$

Description Logics

-

ALC, a basic DL

\mathcal{ALC} , a basic Description Logics

Syntax:

- Ontology: finite set of axioms
- Axioms: $C \equiv D, C \sqsubseteq D, a:C$
- Concepts C, D : $C \sqcap D, C \sqcup D, \neg C$ (full Booleans),
 $\exists r.C, \forall r.C$

Patient \equiv Person $\sqcap \exists \text{suffersFrom.Disease}$

Inflammation \sqsubseteq Disease

HeartDisease \equiv Disease $\sqcap \exists \text{hasLoc.Heart}$

Endocarditis \equiv Inflammation $\sqcap \exists \text{hasLoc.Endocardium}$

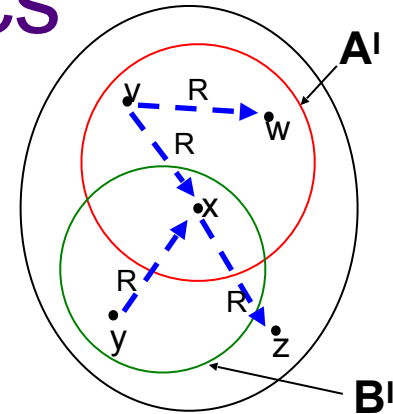
Endocardium \sqsubseteq Bodypart $\sqcap \exists \text{isPartOf.Heart}$

Bob: (Person \sqcap
 $\exists \text{suffersFrom. (Inflammation } \sqcap \exists \text{hasLoc.Endocardium } \sqcap$
 $\forall \text{causedBy.BactInfection))$

\mathcal{ALC} , a basic Description Logics

Semantics: based on interpretations $I = (\Delta, \cdot^I)$ with

- Δ being a non-empty set
- $A^I \subseteq \Delta$
- $r^I \subseteq \Delta \times \Delta$



| Syntax | Interpretation |
|----------------------------|--|
| $Human \sqcap Male$ | $Human^I \cap Male^I$ |
| $Doctor \sqcup Lawyer$ | $Doctor^I \cup Lawyer^I$ |
| $\neg Male$ | $\Delta \setminus Male^I$ |
| $\exists isPartOf.Heart$ | $\{e \in \Delta \mid \text{there is some } f: \\ (e, f) \in isPartOf^I \text{ and } f \in Heart^I\}$ |
| $\forall causedBy.BactInf$ | $\{e \in \Delta \mid \text{for all } f \in \Delta: \text{if} \\ (e, f) \in causedBy^I \text{ then } f \in BactInf^I\}$ |

\mathcal{ALC} , a basic Description Logics

Semantics: based on interpretations $I = (\Delta, \cdot^I)$ with

- Δ being a non-empty set
- $A^I \subseteq \Delta$
- $r^I \subseteq \Delta \times \Delta$

| Axioms | Interpretation: for I to be a model of ...it has to satisfy |
|-------------------|--|
| $A \sqsubseteq B$ | $A^I \subseteq B^I$ |
| $A \equiv B$ | $A^I = B^I$ |
| $b:B$ or $B(b)$ | $b^I \in B^I$ |

Reminder:

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$

for each $\alpha \in O$

Consistency

- a concept A , test whether $O \models A \sqsubseteq \perp$

$A' \neq \emptyset$ in some model I of O

Satisfiability

- two concepts A, B , test whether $O \models A \sqsubseteq B$

$A' \subseteq B'$ in each model I of O

Subsumption

- individual b , concept name A , test whether $O \models b:A$

Instancship

$b \in A'$ in each model I of O

a model of O

Some Exercises

Create an ontology

1. that is consistent.
2. that is not consistent.
3. with an unsatisfiable concept.
4. that entails but does not contain $C \sqsubseteq D$.
5. that entails but does not contain $a:C$.

Reminder:

Given an ontology O

- test whether there is an interpretation I with $I \models \alpha$
for each $\alpha \in O$

Consistency

- a concept A , test whether $O \models A \sqsubseteq \perp$

Satisfiability

- two concepts A, B , test whether $O \models A \sqsubseteq B$

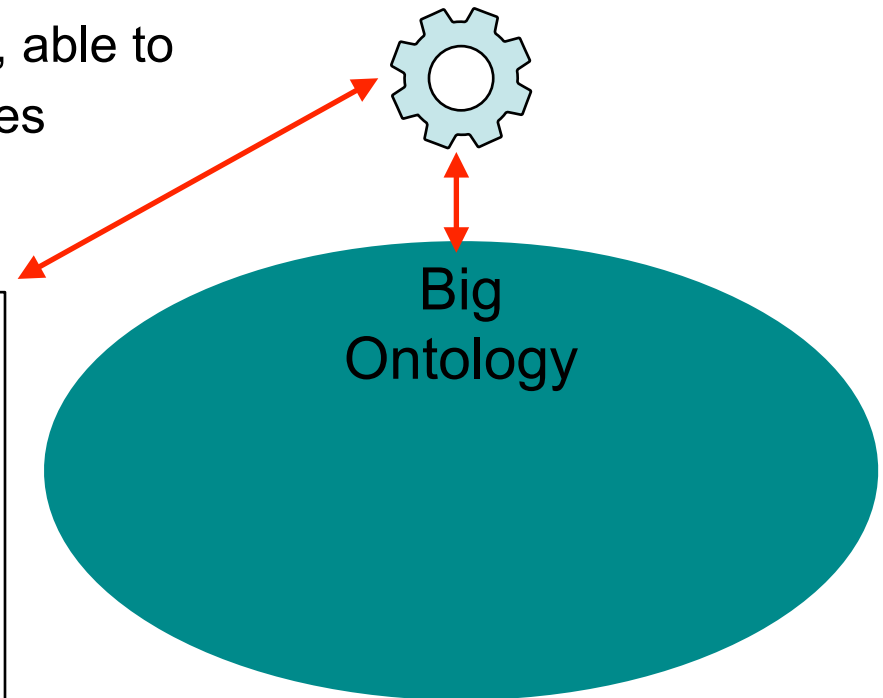
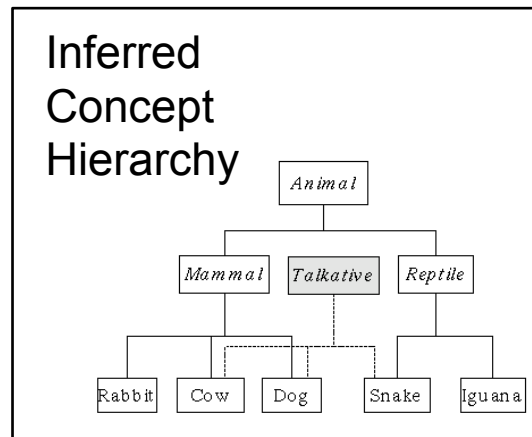
Subsumption

- individual b , concept name A , test whether $O \models b:A$

Instancship

Description Logic Reasoning

- *classification*: involves n^2 entailment tests
- *complexity* of each entailment test $O \models? \dots$ is high
- specialised *DL reasoners* are implemented & optimised
 - ORE reasoner competitions:
 - > 10 top notch reasoners, able to
 - classify complex ontologies with >300K terms



Description Logics Reasoning

- For \mathcal{ALC} ,
all reasoning tasks can be reduced to (in)consistency
➔ we only need a consistency checker 😊

Theorem 2 Let \mathcal{O} be an ontology and a an individual name **not** in \mathcal{O} . Then

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

A Tableau Algorithm for \mathcal{ALC}

- ...similar to *chase*
- a decision procedure for *consistency of \mathcal{ALC} ontologies*
 - sound
 - complete
 - terminating
- takes input ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$
 - transforms all concepts in \mathcal{O} into NNF
 - works on set of ABoxes
 - starts with $\{\mathcal{A}\}$ and applies **tableau rules** until
 - no more rules apply - **complete** - or
 - **clash** occurs
 - returns “ \mathcal{O} is consistent” if complete, clash-free ABox was built
 - this ABox can then be unravelled into an infinite model of input

DeMorgan's law,
duality between
 \forall, \exists

A Tableau Algorithm for \mathcal{ALC}

- \sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$
- \sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ or with $\mathcal{A} \cup \{a : C_2\}$
- \exists -rule: if $a : \exists s.C \in \mathcal{A}$, and there is no b with
 $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$
- \forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$, and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$
- GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, and
if C is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : D\}$
else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

- Observations:
- all rules *add* things
 - \sqcup -rule is *non-deterministic*

A Tableau Algorithm for \mathcal{ALC}

- \sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$
- \sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ or with $\mathcal{A} \cup \{a : C_2\}$
- \exists -rule: if $a : \exists s.C \in \mathcal{A}$, and there is no b with
 $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$
- \forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$, and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$
- GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, and
if C is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : D\}$
else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

- Observations:
- all rules *add* things
 - \sqcup -rule is *non-deterministic*

A Tableau Algorithm for \mathcal{ALC}

Let's work out 2 examples...

\sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$

\sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ or with $\mathcal{A} \cup \{a : C_2\}$

\exists -rule: if $a : \exists s.C \in \mathcal{A}$, and there is no b with
 $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$

\forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$, and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$

GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, and
if C is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : D\}$
else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

- Observations:
- all rules *add* things
 - \sqcup -rule is *non-deterministic*

A Tableau Algorithm for \mathcal{ALC}

\sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$

\sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ or with $\mathcal{A} \cup \{a : C_2\}$

\exists -rule: if $a : \exists s.C \in \mathcal{A}$, and there is no b with
 $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$

\forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$, and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$

GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, and
if C is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : D\}$
else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

Let's work out 2 examples...

Not
terminating



Proper Tableau Rules for \mathcal{ALC}

- \sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$
- \sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ or with $\mathcal{A} \cup \{a : C_2\}$
- \exists -rule: if $a : \exists s.C \in \mathcal{A}$, a is **not blocked**, and there is no b with
 $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$
- \forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$, and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$
- GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, and
if C is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : D\}$
else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

These rules lead to terminating algorithm



Proper Tableau Rules for \mathcal{ALC}

- \sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$
- \sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$, and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ or with $\mathcal{A} \cup \{a : C_2\}$
- \exists -rule: if $a : \exists s.C \in \mathcal{A}$, a is not blocked, and there is no b with $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$
- \forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$, and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$
- GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, and
if C is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : D\}$
else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

Tableau and Reasoning in DLs

This tableau

- is a decision procedure for \mathcal{ALC}
 - can be made to run in PSpace
- is very naive
 - needs serious consideration & optimisation to work
- is only for \mathcal{ALC}
 - needs more rules/blocking for more expressive DLs
- only decides consistency of input ontology
 - classification needs much more & more optimisation
- is only 1 of many DL reasoning techniques
 - consequence-driven
 - hyper-tableau-based

Description Logics & OWL

The Web Ontology Language OWL

Vision of *Semantic Web* led to web ontology languages
W3C set up Web-Ontology Working Groups



E. Shepard,
Winnie-the-Pooh [A. A. Milne]



The Web Ontology Language OWL

OWL 2 is a *W3C recommendation*

- based on a DL more expressive than \mathcal{ALC}
 - inverse roles, cardinality restrictions, role (chain) inclusions,...
 - datatypes (e.g., integers, strings) for features (e.g., *weight*, *age*, *name*)
- where axioms, concepts, etc. can be *annotated*
 - who wrote axiom? Who introduced a term?
 - labels, synonyms, preferred labels, different languages, etc.
- with an *imports* mechanism for modular development/reuse
- with a *versioning* mechanism
- in different syntaxes
- with 3 fragments/profiles for more efficient reasoning
 - each optimal for complexity class

What OWL has changed

Having a stable, **standardized** syntax

⇒ more **tools**:

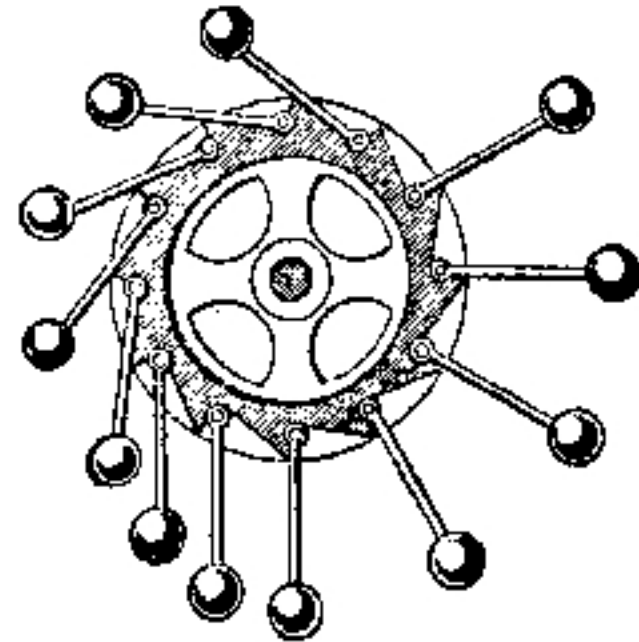
- reasoners
- APIs
- editors/IDEs

⇒ more **users** designing more **ontologies**:

- enthusiasm around Semantic Web
- requirements in bio-health applications
- see e.g. BioPortal repository: 621 OWL ontologies

⇒ more **requirements**:

- (non-logician) domain experts require support to build/maintain/use large scale, highly axiomatised ontologies
- performance/scalability
- novel reasoning problems



Explanations of Entailments

-

demo and brief tour

Explanation of Entailments



- Entailments are either
 - intended or
 - reveal **modelling errors** ➡ which require repair
 - have to be understood and **explained**

Two approaches to Explanation

1. show **proofs**:

- pick suitable calculus
- present proof steps

Two approaches to Explanation

1. show **proofs**:

- pick suitable calculus
- present proof steps

2. show **justifications**:

- Let $\mathcal{O} \models \alpha$.

$\mathcal{J} \subseteq \mathcal{O}$ is a **justification** for α

- $\mathcal{J} \models \alpha$ and
- \mathcal{J} is minimal

- ▶ also called *kernels*
- ▶ 1 entailment can have several entailments
- ▶ to repair α ,
weaken each justification
- ▶ to trust α ,
understand easiest justification

Two approaches to Explanation

1. show **proofs**:

- pick suitable calculus
- present proof steps
- approach often followed

2. show **justifications**:

- Let $\mathcal{O} \models \alpha$.

$\mathcal{J} \subseteq \mathcal{O}$ is a **justification** for α

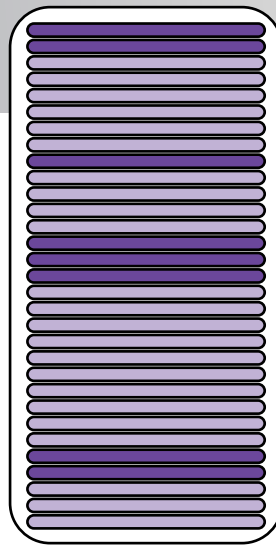
- $\mathcal{J} \models \alpha$ and
- \mathcal{J} is minimal

► also called *kernels*

► 1 entailment can have several entailments

require...

| | | |
|---|---|---|
| X | user to understand calculus & steps | |
| X | modifications to reasoner to produce proofs | |
| X | suitable summarization & choice of proof | |
| X | user to be happy with | X |



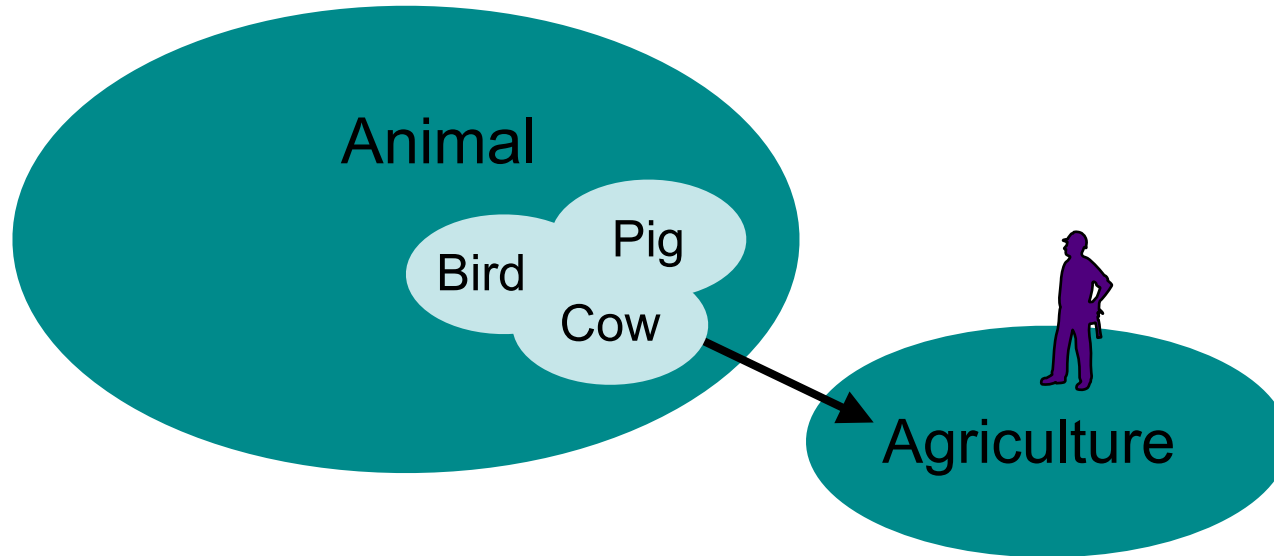
Justifications

- Reveal needle in haystack
- How can we compute them?
 - Glass-box: add tracing mechanism to reasoner
 - Black-box: use Reiter's hitting sets
 - interesting:
 - black-box as good as glass-box
 - 'true' entailments are easier than $\mathcal{O} \models \top \sqsubseteq \perp$
- Check out in Protégé
 - ➡ demo
- Justifications can be refined:
 - *lemmatise*
 - remove superfluous axiom parts: laconic
- Explaining 1 entailment is not enough:
 - imagine reasoner finds 144/395 classes are **unsatisfiable**

Modularity

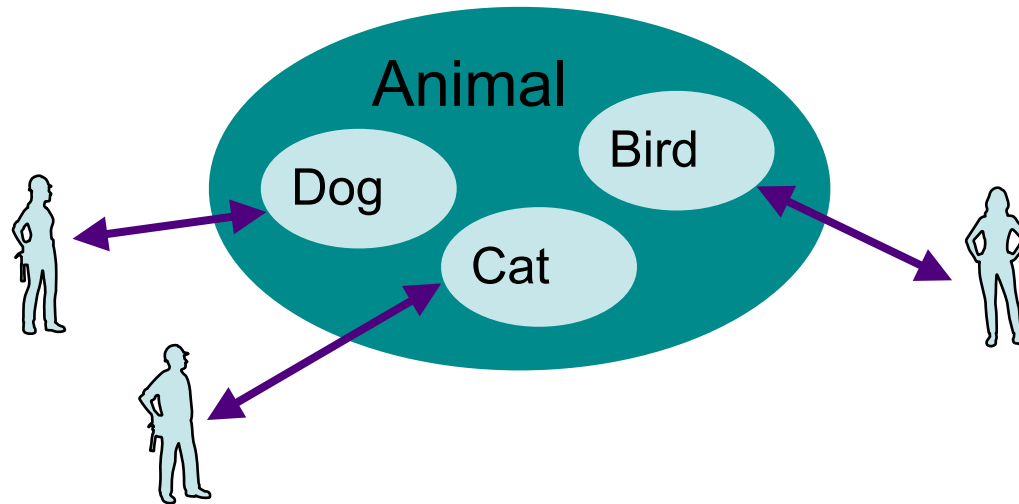
Why Modularity?

- for re-using terms from existing ontologies



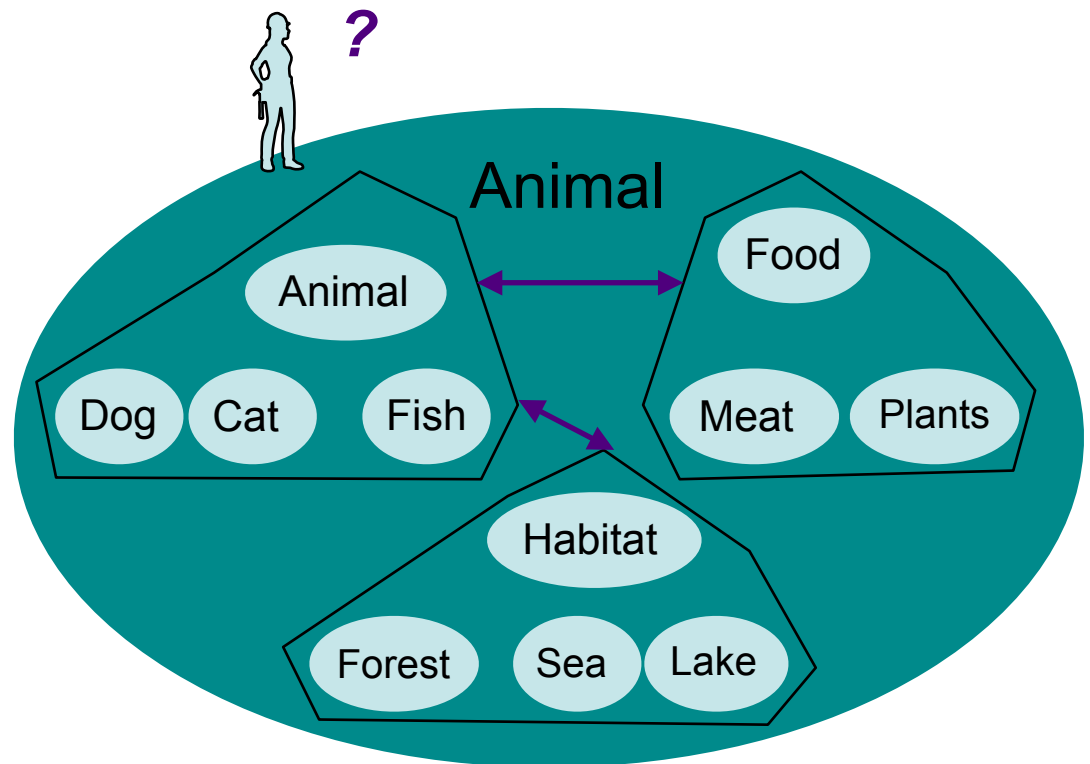
Why Modularity?

- for re-using terms from existing ontologies
- collaboratively working on an ontology



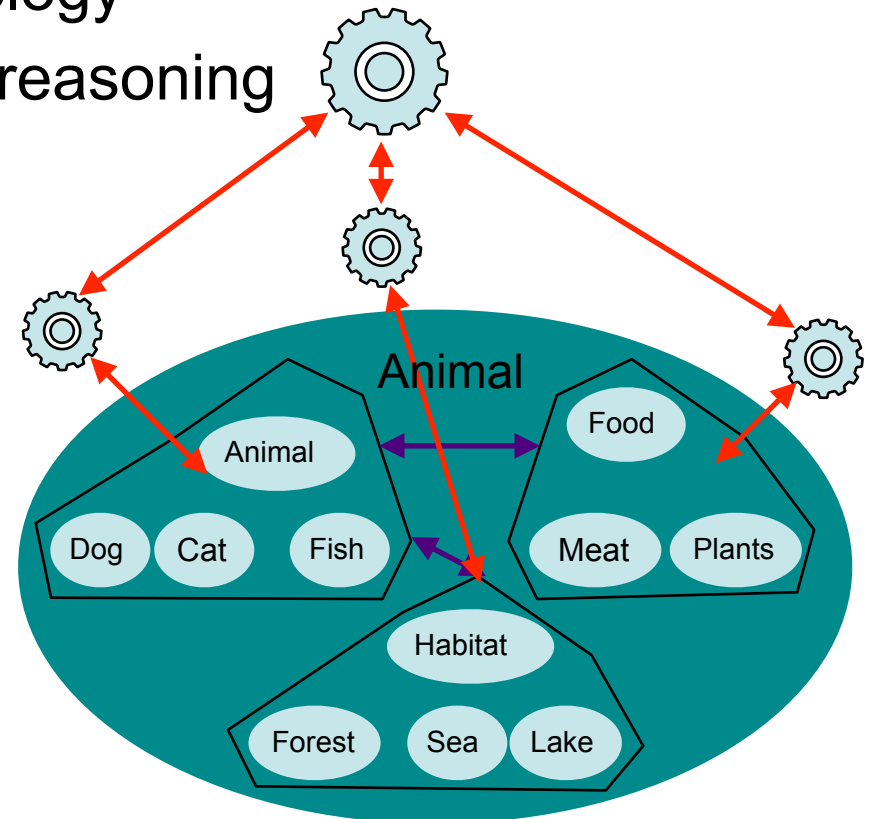
Why Modularity?

- for re-using terms from existing ontologies
- collaboratively working on an ontology
- understanding an ontology, its
 - content
 - modelling
 - structure



Why Modularity?

- for re-using terms from existing ontologies
- collaboratively working on an ontology
- understanding an ontology
- optimising automated reasoning
 - classification



A Module

- is a *subset* of an ontology
- that *covers* a set of terms

A Module

- is a *subset* of an ontology
- that *covers* a set of terms

Given an ontology \mathcal{O} ,
a signature Σ ,
a Σ -module \mathcal{M} of \mathcal{O} is

- a subset of \mathcal{O}
- that is deductively indistinguishable from \mathcal{O} , i.e.,
 - for all α over Σ

$$\mathcal{O} \models \alpha \text{ iff } \mathcal{M} \models \alpha$$

A Module

- is a *subset* of an ontology
- that *covers* a set of terms

Given an ontology \mathcal{O} ,
a signature Σ ,
a Σ -module \mathcal{M} of \mathcal{O} is

- a subset of \mathcal{O}
- that is deductively indistinguishable from \mathcal{O} , i.e.,
 - for all α over Σ

$$\mathcal{O} \models \alpha \text{ iff } \mathcal{M} \models \alpha$$

In which
logic?

A Module

- is a *subset* of an ontology
- that *covers* a set of terms

Any? Minimal?
Good?

Given an ontology \mathcal{O} ,
a signature Σ ,
a Σ -module \mathcal{M} of \mathcal{O} is

- a subset of \mathcal{O}
- that is deductively indistinguishable from \mathcal{O} , i.e.,
 - for all α over Σ

$$\mathcal{O} \models \alpha \text{ iff } \mathcal{M} \models \alpha$$

In which
logic?

A Module

- is a *subset* of an ontology
- that *covers* a set of terms

Any? Minimal?
Good?

Conservative
extensions

Given an ontology \mathcal{O} ,
a signature Σ ,
a Σ -module \mathcal{M} of \mathcal{O} is

- a subset of \mathcal{O}
- that is deductively indistinguishable from \mathcal{O} , i.e.,
 - for all α over Σ

$$\mathcal{O} \models \alpha \text{ iff } \mathcal{M} \models \alpha$$

In which
logic?

A Module

- is a *subset* of an ontology
- that *covers* a set of terms

Any? Minimal?
Good?

Conservative
extensions

Given an ontology \mathcal{O} ,
a signature Σ ,
a Σ -module \mathcal{M} of \mathcal{O} is

- a subset of \mathcal{O}
- that is deductively indistinguishable from \mathcal{O} , i.e.,
 - for all α over Σ

$$\mathcal{O} \models \alpha \text{ iff } \mathcal{M} \models \alpha$$

In which
logic?

How to test/
extract?

Properties of Good Modules

1. realisable

- decidable
- low complexity

Properties of Good Modules

1. realisable
2. unique

Properties of Good Modules

1. realisable
2. unique
3. small (?)

Properties of Good Modules

1. realisable
2. unique
3. small (?)
4. good logical properties

Properties of Good Modules

1. realisable
2. unique
3. small (?)
4. good logical properties
 - self-contained:

M and O have same entailments regarding $\Sigma \cup \text{sign}(M)$

Otherwise:
2 kinds of terms
(covered & not)

Properties of Good Modules

1. realisable
2. unique
3. small (?)
4. good logical properties

- self-contained:

M and O have same entailments regarding $\Sigma \cup \text{sign}(M)$

- depleting:

OM has no (real) entailments regarding $\Sigma \cup \text{sign}(M)$

Encapsulation!

Properties of Good Modules

1. realisable
2. unique
3. small (?)
4. good logical properties

- self-contained:

M and O have same entailments regarding $\Sigma \cup \text{sign}(M)$

- depleting:

OM has no (real) entailments regarding $\Sigma \cup \text{sign}(M)$

- robust for certain operations:

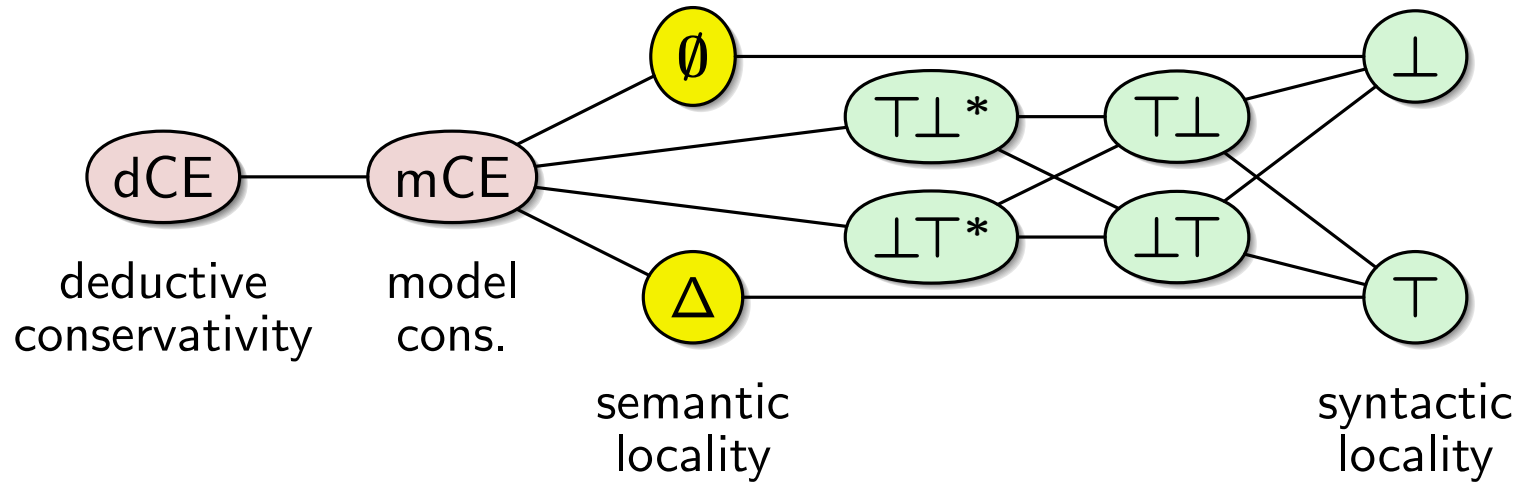
- vocabulary restriction:

M is also a Σ' -module of O for $\Sigma' \subseteq \Sigma$

- ...

Rational

Some Relevant Types of Modules



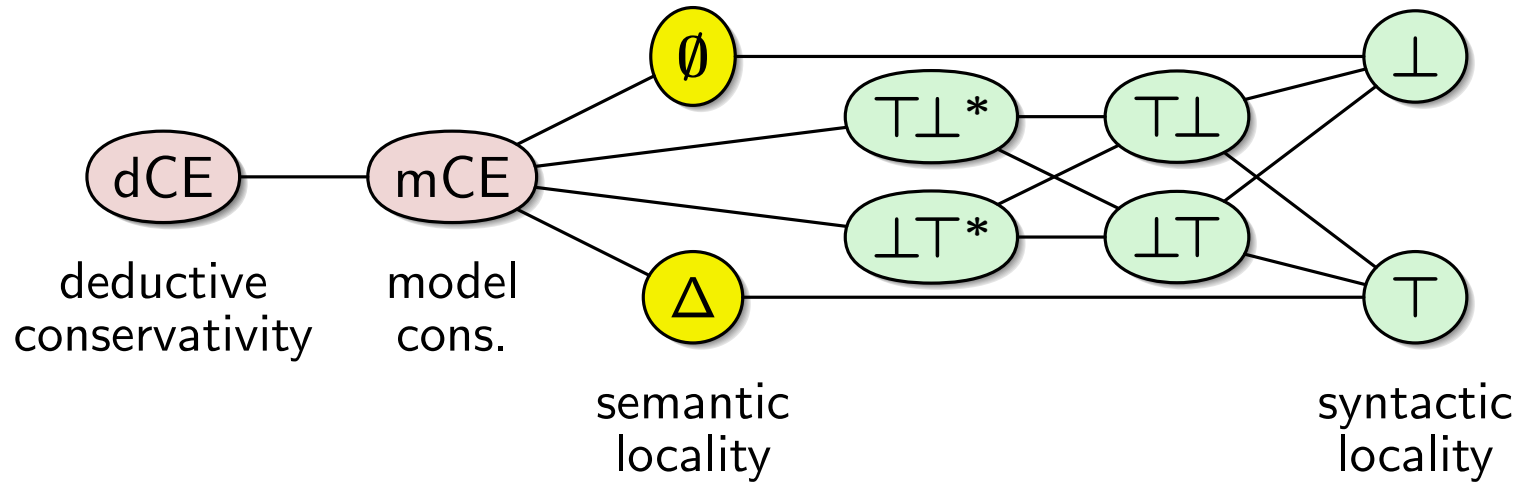
$$x \text{ --- } y \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

intractable ... undecidable

as difficult as reasoning

tractable

Some Relevant Types of Modules



$$x \text{ --- } y \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

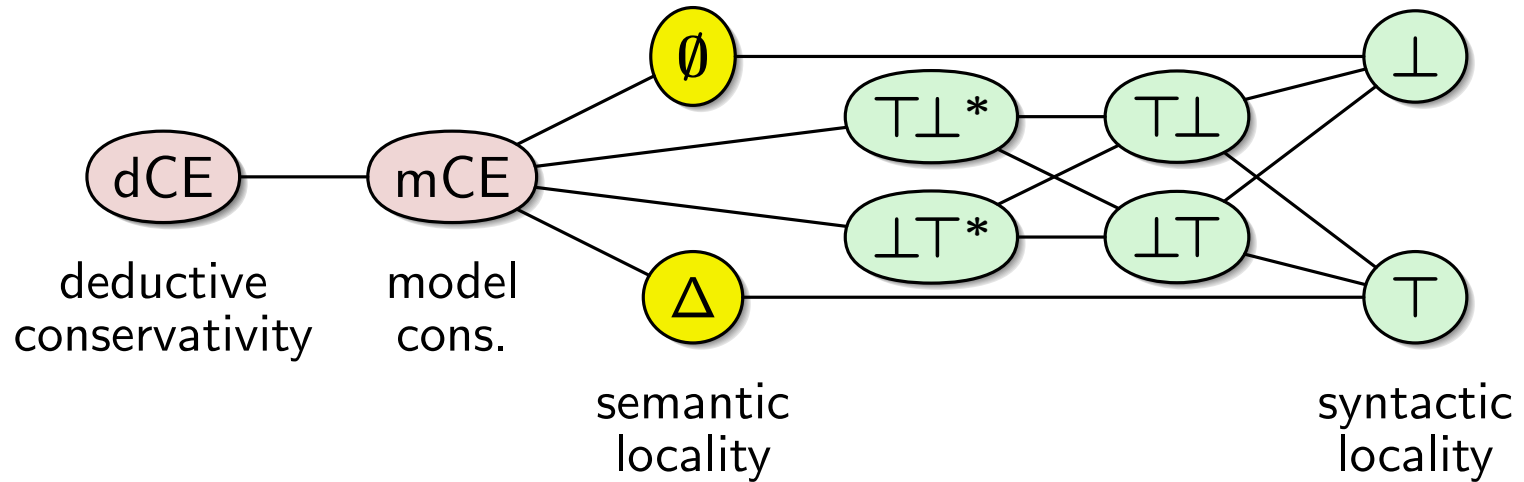
○ intractable ... undecidable

● as difficult as reasoning

○ tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Some Relevant Types of Modules



small \subseteq large

$$(x) - (y) \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

○ intractable ... undecidable

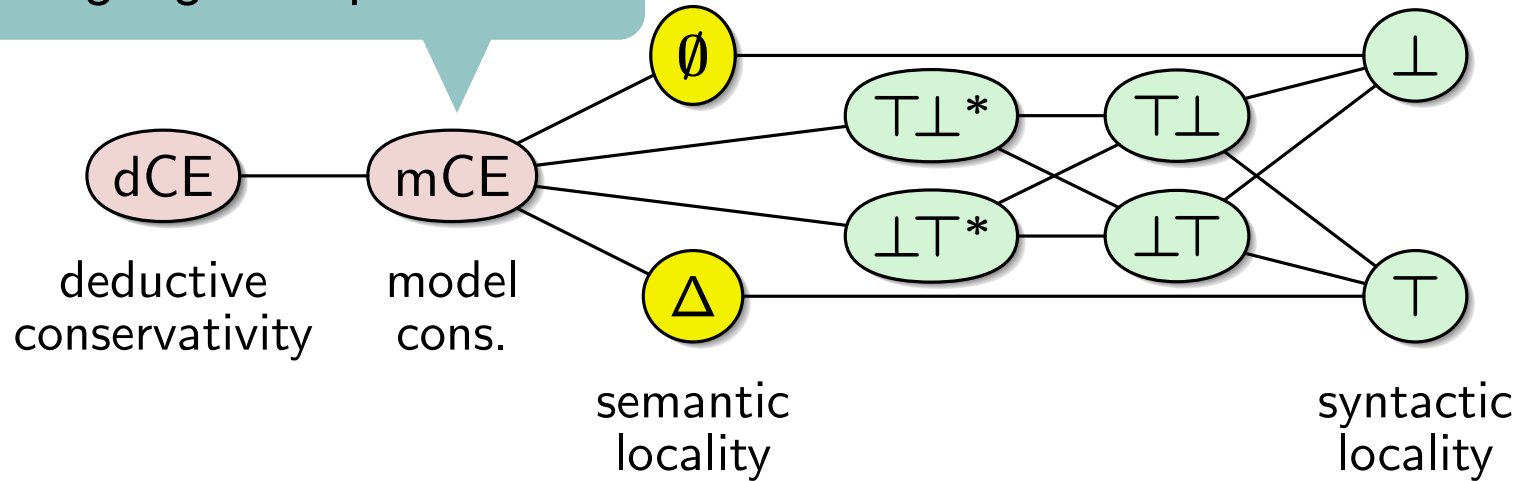
● as difficult as reasoning

○ tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Some Relevant Types of Modules

Nice: language independent!



$$x - y \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

intractable ... undecidable

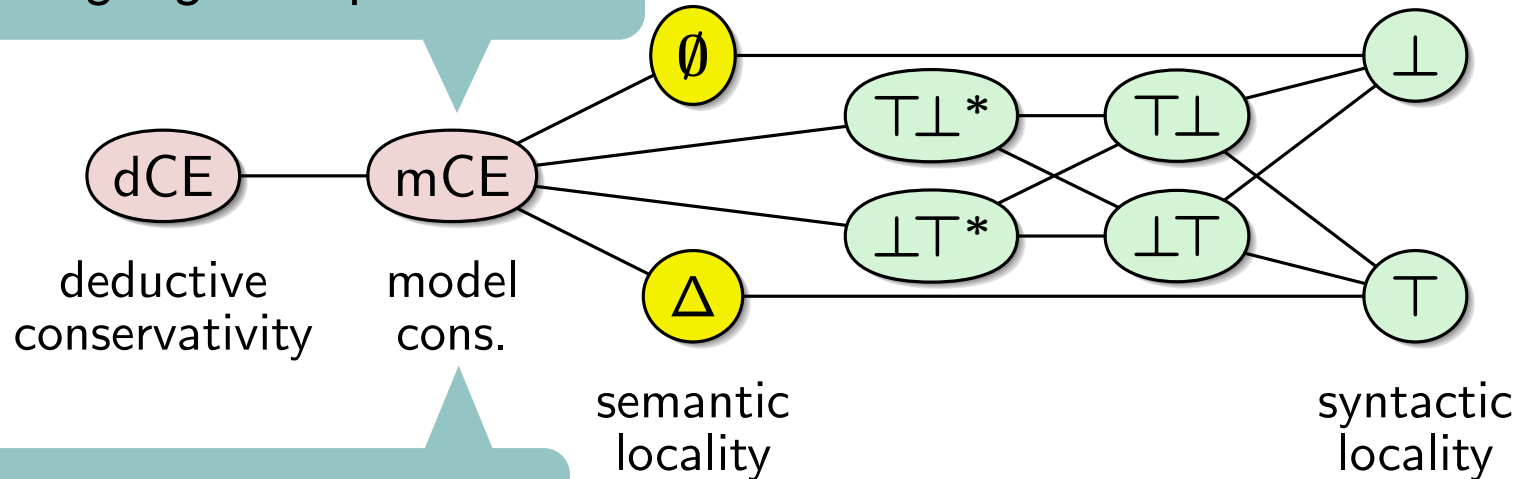
as difficult as reasoning

tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Some Relevant Types of Modules

Nice: language independent!



- undecidable for EL, acyclic ALC TBoxes...
- PT—NExpTime-c for tiny DLs

$$(x) - (y) \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

○ intractable ... undecidable

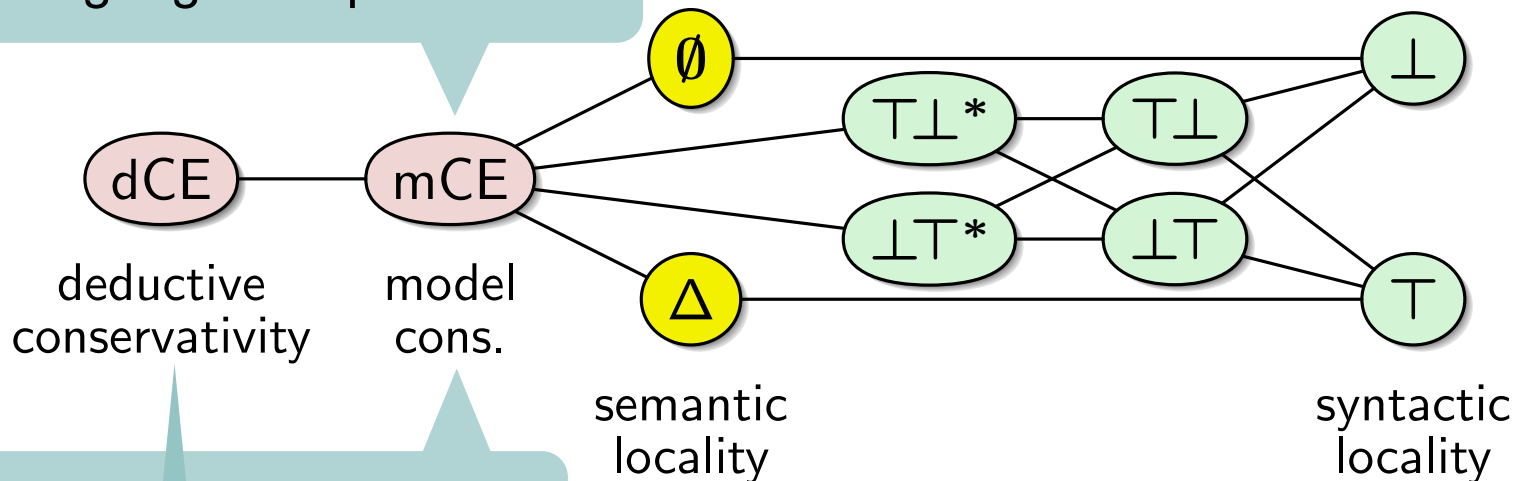
● as difficult as reasoning

○ tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Some Relevant Types of Modules

Nice: language independent!



- undecidable for EL, acyclic ALC TBoxes...
- PT—NExpTime-c for tiny DLs

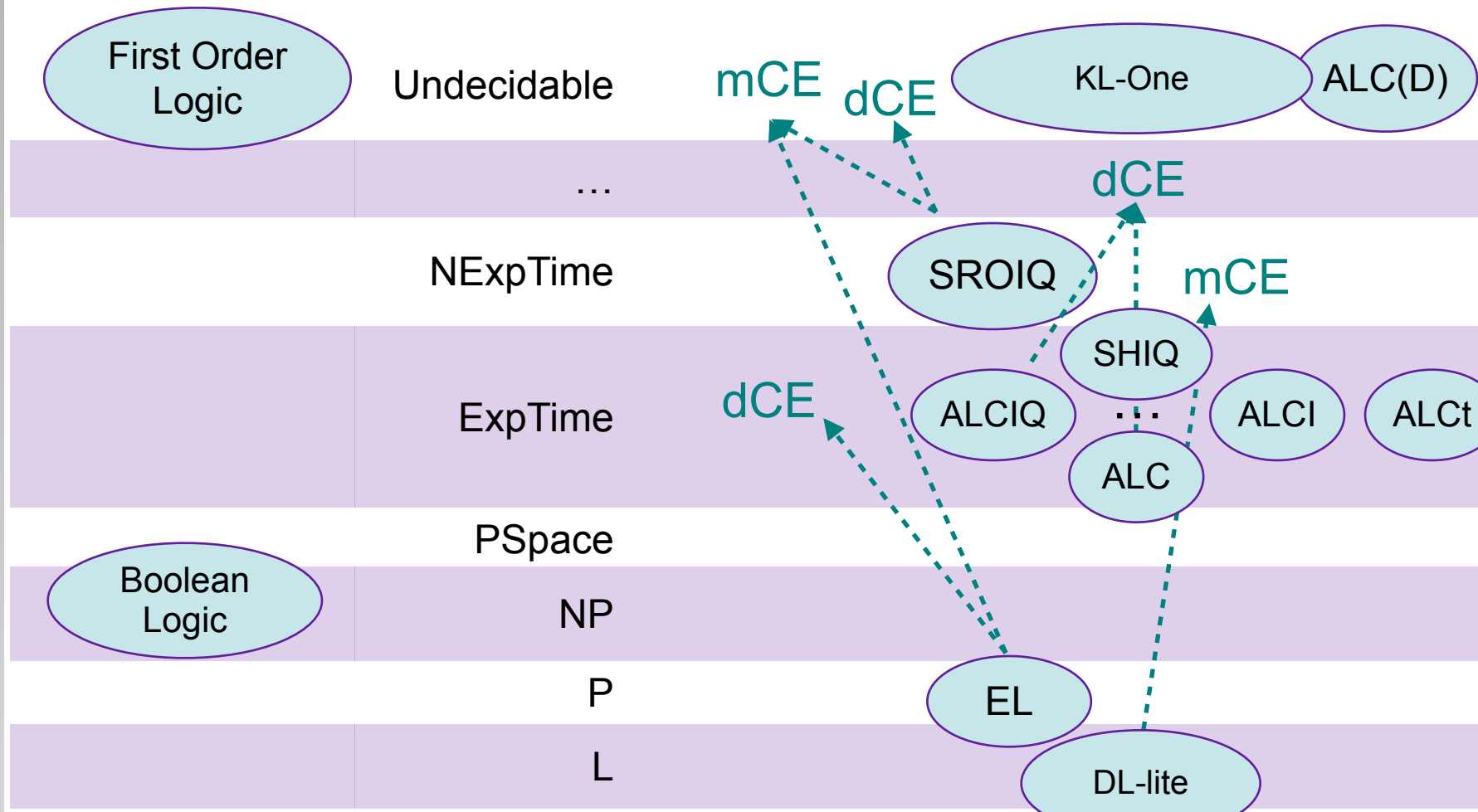
- ExpTime-c for EL
- 2ExpTime-c for ALC - ALCQI
- undecidable for ALCQIO and thus OWL

$$x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

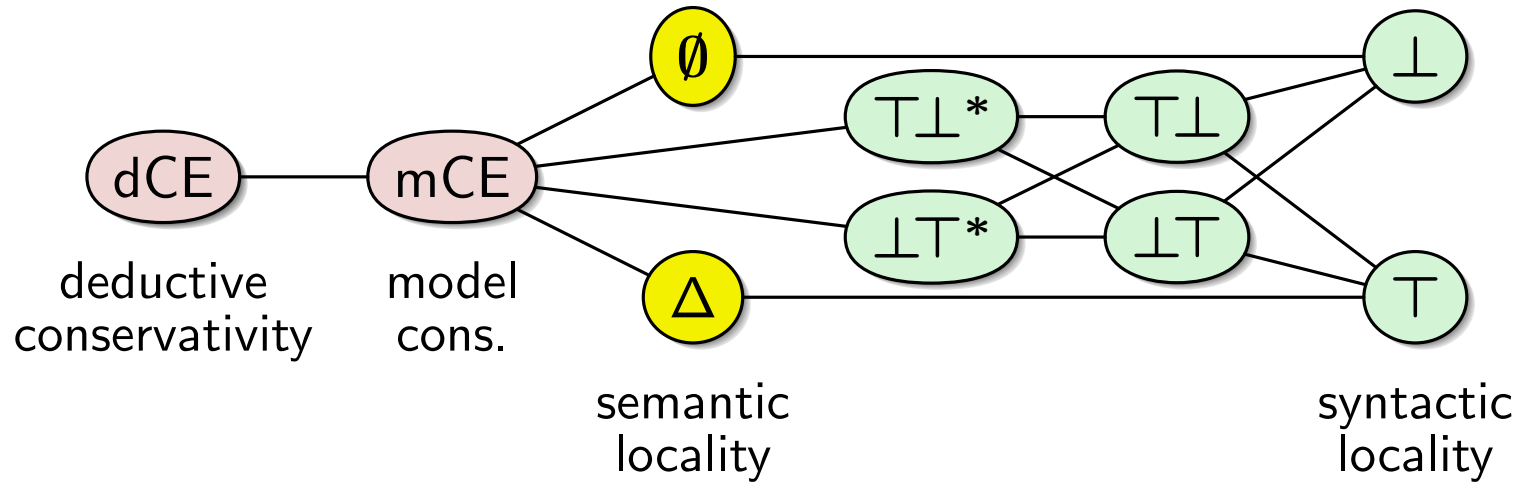
- intractable ... undecidable
- as difficult as reasoning
- tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Complexity of Entailment Checking versus Conservativity Checking



Some Relevant Types of Modules



$$(x) - (y) \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

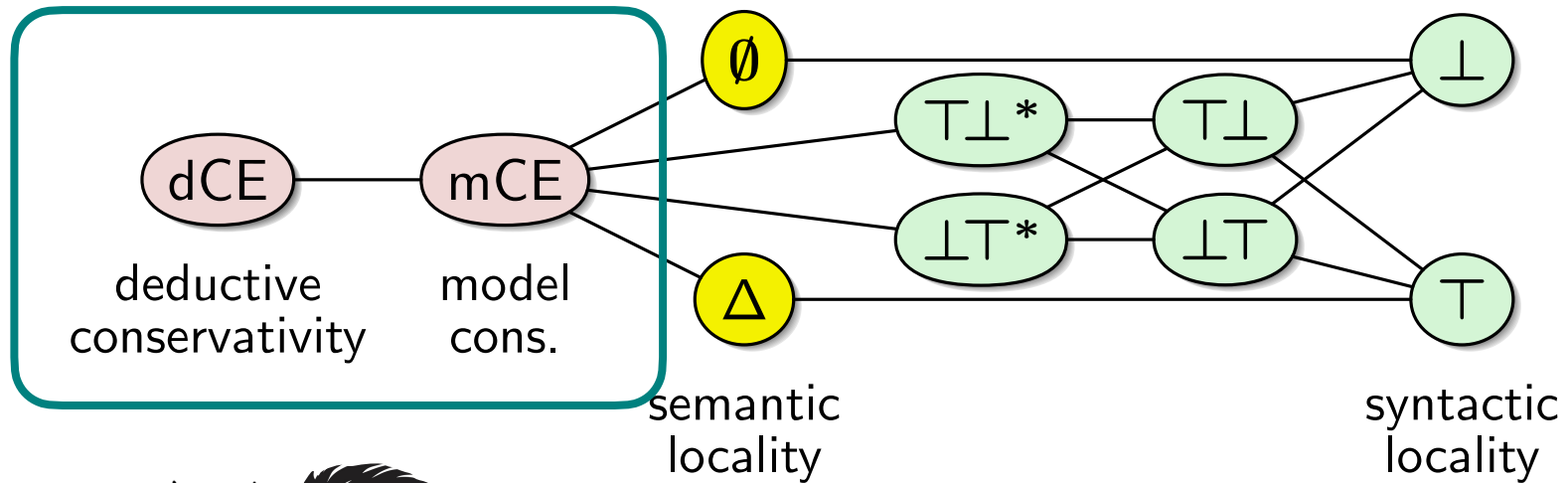
○ intractable ... undecidable

○ as difficult as reasoning

○ tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Some Relevant Types of Modules



$$x - y \quad x\text{-module}(\mathcal{O}, \Sigma) \subseteq y\text{-module}(\mathcal{O}, \Sigma)$$

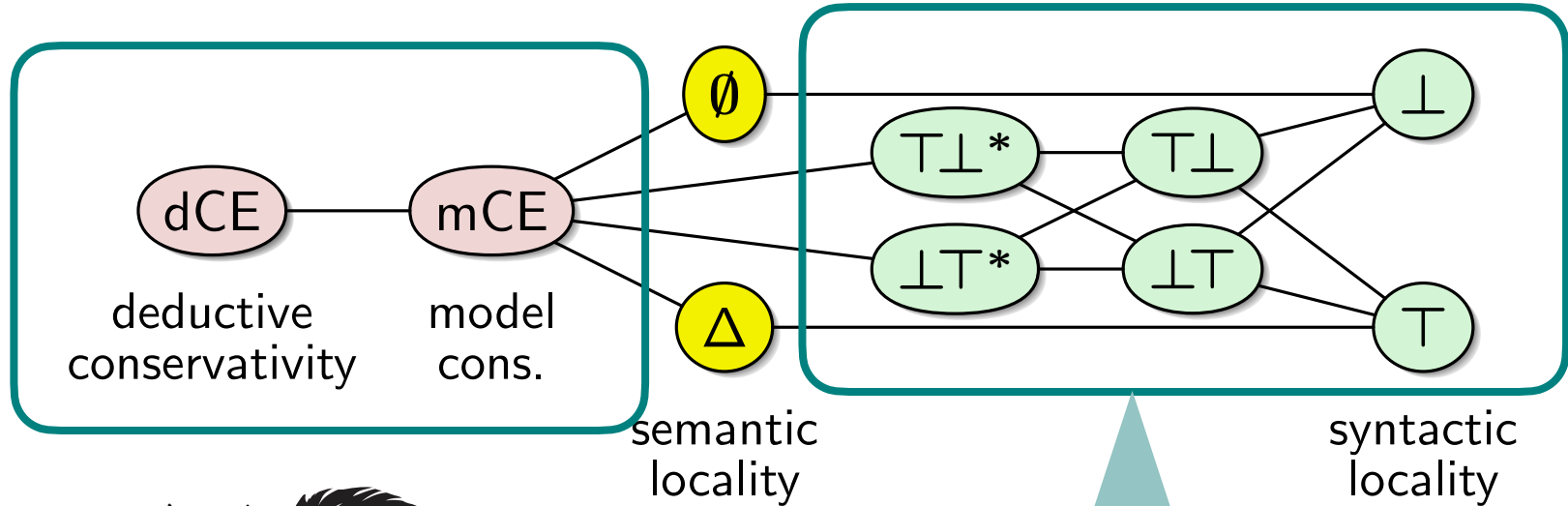
○ intractable ... undecidable

○ as difficult as reasoning

○ tractable

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

Some Relevant Types of Modules



My favourite:

- ✓ language-independent
- ✓ polynomial
- ✓ depleting
- ✓ self-contained
- ✓ unique
- not minimal
- sometimes huge

Module(\mathcal{O}, Σ)

B. Cuenca Grau
S. Ghilardi
C. Lutz
F. Wolter
...

The End

- A brief tour through various aspects of DLs
 - including tableau, modules, and explanations
- What was left out
 - details, proofs, complexity bounds
 - implementation and optimisations
 - many extensions & restrictions, *EL*, *SHIQ*, *SROIQ*, ...
 - variations: temporal, non-monotonic, fuzzy, metric, ...
 - reasoning problems:
 - conjunctive query answering
 - rewriting, msc, lcs, unification, matching,
 - decomposition
 - alignment, diffing

Thank You!

