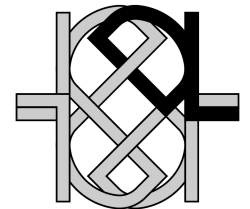
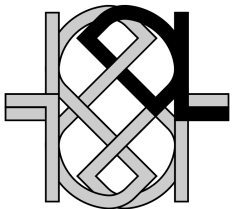


# Reasoning in Description Logics: Expressive Power vs. Computational Complexity

Carsten Lutz

University of Bremen, Germany



## Motivation

Description Logic is subfield of KR concerned with **terminological knowledge**:

Describe the **central notions** of the application domain (its **terminology**)  
and their **interrelations**

E.g. in medical applications:

**Tissue, Inflammation, Pericadium, Pericarditis**, etc.

DLs play important role as **logical foundation of ontology languages**:

- OWL is the **W3C-standard for a Web Ontology Language**

OWL 1 in 2004

OWL 2 in 10/2009

- OWL is essentially a **description logic with an XML syntax**



## Motivation

Main reason for popularity:

**attractive compromise** between expressive power and computational complexity

### Propositional Logic

Efficient reasoning via SAT solvers, but often too inexpressive

### First-Order Logic

Very expressive reference formalism, but reasoning too costly

≈ Modal  
Logic

Not one DL, but a large **toolbox of formalisms**:

- DLs cover **broad range** of responses to “complexity vs. expressive power”
- OWL 2 contains different **profiles** (3 inexpressive, 1 expressive, 1 not a logic)



Before break:

- brief introduction to description logics
- complexity and expressive power of expressive DLs
- complexity and expressive power of lightweight DLs, part I

After break:

- complexity and expressive power of lightweight DLs, part II
- instance data and query answering



# Introduction to Description Logics



## Some DL Basics

Knowledge is (mainly) stored in the **TBox**, e.g.:

$$\text{Pericardium} \sqsubseteq \text{Tissue} \sqcap \exists \text{partOf.Heart}$$
$$\text{Pericarditis} \doteq \text{Inflammation} \sqcap \exists \text{location.Pericardium}$$
$$\text{Inflammation} \sqsubseteq \text{Disease} \sqcap \exists \text{actsOn.Tissue}$$
$$\text{Tissue} \sqcap \text{Disease} \sqsubseteq \perp$$

TBox = “Terminology Box”; modern view: **TBox = ontology**

Formally, a TBox is a finite set of

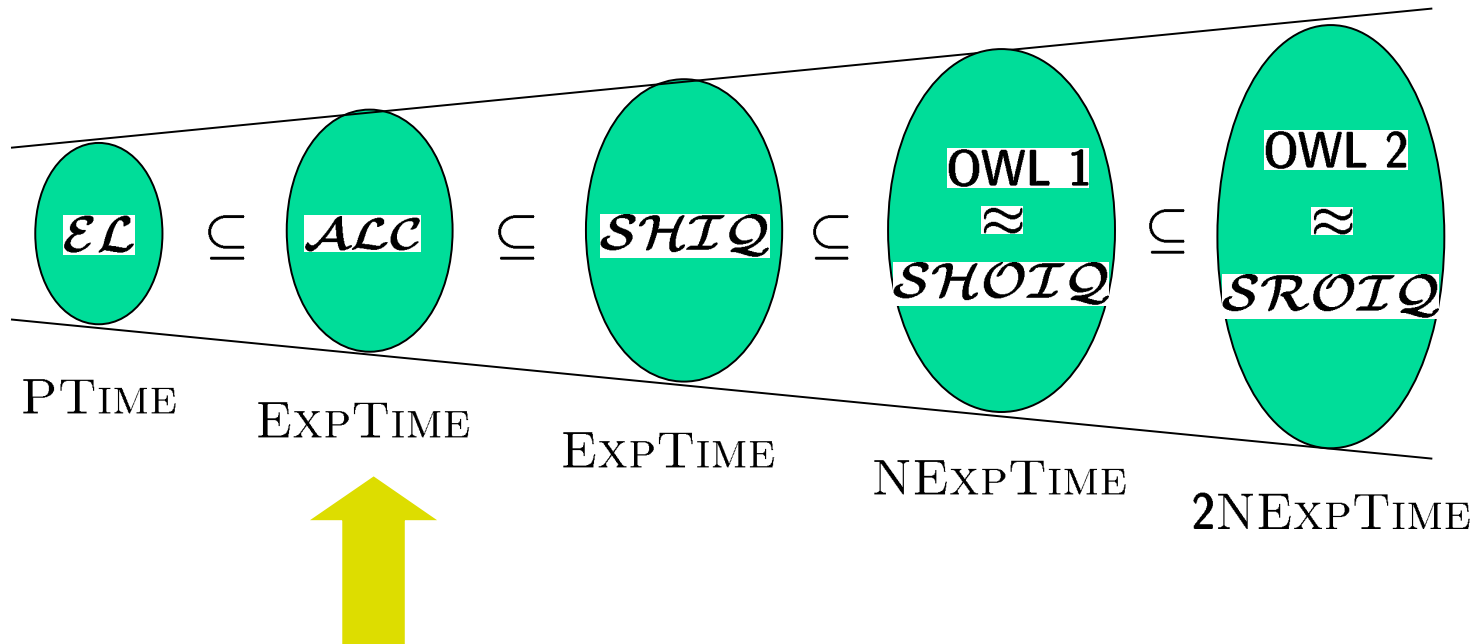
**concept inclusions**  $C \sqsubseteq D$     and    **concept definitions**  $C \doteq D$

where  $C, D$  are **concepts** ( $\approx$  formulas) in the DL used.



## Some DL Basics

Different concept constructors give rise to different DLs / OWL dialects:



## The Description Logic $\mathcal{ALC}$

Fix a countably infinite supply of

• **concept names** ( $\sim$  unary predicates)

• **role names** ( $\sim$  binary predicates)

Concept language of  $\mathcal{ALC}$ :

$$C ::= A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C$$

$\exists r.C$ : existential restriction

$\forall r.C$ : universal restriction / value restriction

For example:

$\text{Disease} \sqcap \exists \text{actsOn. Organ} \sqcap \forall \text{cause. } \neg \text{Genetic}$





DL interpretation  $\mathcal{I}$ :

FO structure with only unary+binary predicates = Kripke structure

DL-style notation: interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  with

- $\Delta^{\mathcal{I}}$  a non-empty set, the **domain**
- $\cdot^{\mathcal{I}}$  the **interpretation function** which assigns
  - a set  $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$  to each concept name  $A$
  - a binary relation  $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$  to each role name  $r$

We now extend  $\cdot^{\mathcal{I}}$  to composite concepts



## Semantics

DL concepts  $\approx$  FO formulas with exactly 1 free variable  $\approx$  modal formulas

$A$	$A(x)$	$p_A$
$\neg C$	$\neg C(x)$	$\neg C$
$C \sqcup D$	$C(x) \vee D(x)$	$C \vee D$
$C \sqcap D$	$C(x) \wedge D(x)$	$C \wedge D$
$\exists r.C$	$\exists y.( r(x, y) \wedge C(y) )$	$\langle r \rangle.C$
$\forall r.C$	$\forall y.( r(x, y) \rightarrow C(y) )$	$[r].C$

Note: 2 variables / guarded formulas suffices



We use  $C^{\mathcal{I}}$  to denote the set  $\{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models C(d)\}$

TBoxes correspond to **FO** sentences:

$$C \sqsubseteq D \qquad \forall x. ( C(x) \rightarrow D(x) )$$

$$C \doteq D \qquad \forall x. ( C(x) \leftrightarrow D(x) )$$

$$\mathcal{T} \qquad \bigwedge_{\varphi \in \mathcal{T}} \varphi$$

Example:

$$\text{Pericardium} \sqsubseteq \text{Tissue} \sqcap \exists \text{partOf.Heart}$$

translates to

$$\forall x. ( \text{Pericardium}(x) \rightarrow ( \text{Tissue}(x) \wedge \exists y. ( \text{partOf}(x, y) \wedge \text{Heart}(y) ) ) )$$



Traditional reasoning problems:

- **satisfiability**: given  $C$  and  $\mathcal{T}$ , is there a model  $\mathcal{I}$  of  $\mathcal{T}$  with  $C^{\mathcal{I}} \neq \emptyset$ ?

used for detecting **modelling mistakes**

- **subsumption**: given  $C$ ,  $D$  and  $\mathcal{T}$ , does  $\mathcal{T} \models C \sqsubseteq D$ ?

i.e., do all models  $\mathcal{I}$  of  $\mathcal{T}$  satisfy  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ?

used to arrange all concepts in a TBox in a **subsumption hierarchy**

makes structure explicit, facilitates **browsing** and **navigation**



Doors Protégé 3.2 beta (file:/home/ast/turhan/GonMoRe/Ontologies/Final-Tests/Door-Tests/Doors.pprj, OWL / RDF Files)

File Edit Project OWL Code Tools Window Help

● OWLClasses ■ Properties ■ Forms ◆ Individuals ◆ Metadata ◆ OWL-DL Individuals ● OWLViz

SUBCLASS EXPLORER

For Project: ● Doors

Asserted Hierarchy

- owl:Thing
  - Activity
  - AuthorizedPerson
  - Context
  - DeliveryGoods
  - Device
    - AudioEnabledDevice
      - AudioEnabledConnectedDevice
    - InstantMessageEnabledDevice
      - InstantMessageEnabledConnectedDevice
    - MobileDevice
      - Handy
      - Laptop
      - PDA
      - SmartPhone
    - SMSEnabledDevice
    - StationaryDevice
    - TextEnabledConnectedDevice
    - TextEnabledDevice
    - VideoEnabledConnectedDevice
    - VideoEnabledDevice
  - Location
  - Network
  - Person
  - Presence
  - ValuePartition

SUBCLASS EXPLORER

For Project: ● Doors

Inferred Hierarchy

- owl:Thing
  - Activity
  - Context
  - DeliveryGoods
  - Device
    - AudioEnabledDevice
      - AudioEnabledConnectedDevice
      - DoorBell
    - TextEnabledDevice
  - MobileDevice
    - Handy
    - InstantMessageEnabledDevice
      - InstantMessageEnabledConnectedDevice
    - Laptop
    - PDA
    - SmartPhone
  - StationaryDevice
  - VideoEnabledDevice
- Location
- Network
- Person
- ValuePartition

CLASS EDITOR

For Class: InstantMessageEnabledConnectedDevice

Name

InstantMessageEnabledConnectedDevice

Annotations

Property	Value	La...
rdfs:...		

Asserted Conditions

InstantMessageEnabledDevice

isConnectedVia some Network

Laptop or PDA [from InstantMessageEnabledDevice]

Properties

- isConnectedVia (multiple Network)
  - Network
- isOwnedBy (multiple Person)
- isUsedBy (single Person)
- supportsLink (multiple Network)

Dis joints

Logic View Properties View

Class

Changed direct superclasses

- AudioReachableInUrgencyContext
- AudioReachableResidentContext
- AuthorisedPersonRingingContext
- AuthorizedNeighbourRingingContext
- AuthorizedPerson
- Building
- ConnectedResident
- DoorBell
- DoorbellReachableInUrgencyContext
- DoorbellReachableResidentContext
- DoorContext
- Handy
- InstantMessageEnabledConnectedDevice
- InstantMessageEnabledDevice

- Removed PresenceContext
- Removed ReachableContext
- Added PresenceContext
- Moved from DoorContext to AuthorisedPersonRingingContext
- Moved from owl:Thing to Person
- Removed Location
- Added ConnectedPerson
- Added AudioEnabledDevice
- Removed PresenceContext
- Removed ReachableContext
- Moved from Context to ConnectedContext, PresenceContext
- Added SMSEnabledDevice
- Added TextEnabledConnectedDevice
- Moved from Device to TextEnabledDevice, VideoEnabledDevice, MobileDevice

Classification Results

Traditional reasoning problems:

- **satisfiability**: given  $C$  and  $\mathcal{T}$ , is there a model  $\mathcal{I}$  of  $\mathcal{T}$  with  $C^{\mathcal{I}} \neq \emptyset$ ?

used for detecting **modelling mistakes**

- **subsumption**: given  $C$ ,  $D$  and  $\mathcal{T}$ , does  $\mathcal{T} \models C \sqsubseteq D$ ?

i.e., do all models  $\mathcal{I}$  of  $\mathcal{T}$  satisfy  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ?

used to arrange all concepts in a TBox in a **subsumption hierarchy**

makes structure explicit, facilitates **browsing** and **navigation**

Note: ■  $C$  satisfiable w.r.t.  $\mathcal{T}$  iff  $\mathcal{T} \not\models C \sqsubseteq \perp$

■  $\mathcal{T} \models C \sqsubseteq D$  iff  $C \sqcap \neg D$  unsatisfiable w.r.t.  $\mathcal{T}$



## On the Role of Complexity

Is DL all about computational complexity?

What complexity theory can do for us:

- help to **understand the expressive power** of the formalism  
to prove **hardness results**, one must show that something **can be expressed**
- provide **performance guarantees** or show that they do not exist

What it cannot do for us (so far):

- tell us whether something will **work in practice** or not



Expressive Description Logics  
(i.e.:  $\mathcal{ALC}$  and above)





## A Bit of History

Stone age of description logics (until mid-1990ies):

“We have to offer efficient reasoning and thus cannot include all Booleans”

“Every application needs at least conjunction and universal restriction”

(and thus reasoning is co-NP-complete)

The *SHIQ* era (since mid-1990ies):

“ExpTime DLs can be implemented efficiently” (FaCT system by Horrocks)

“We do need the Booleans and much, much more (but want to stay decidable)!”



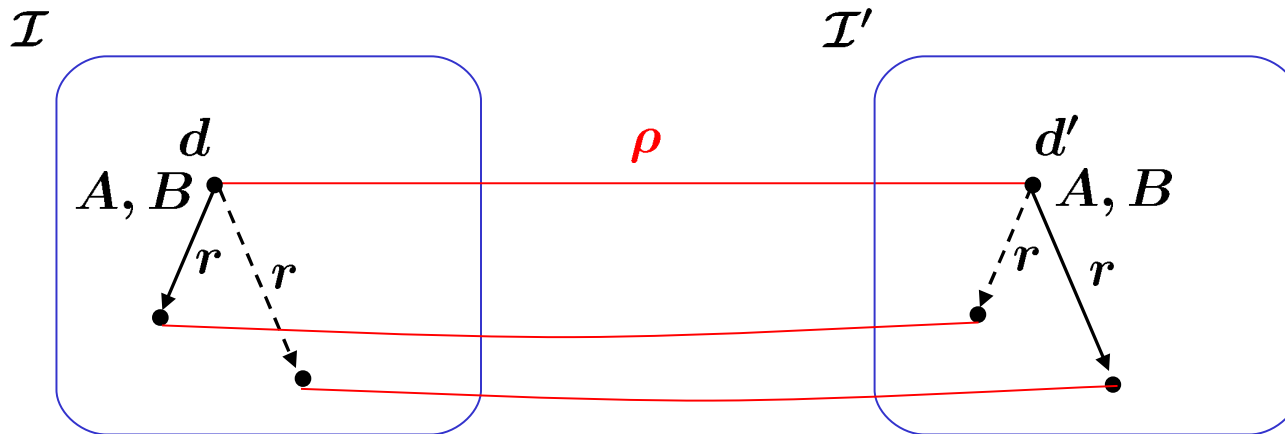
## Expressive Power of $\mathcal{ALC}$

Central notion for understanding expressive power of  $\mathcal{ALC}$ :

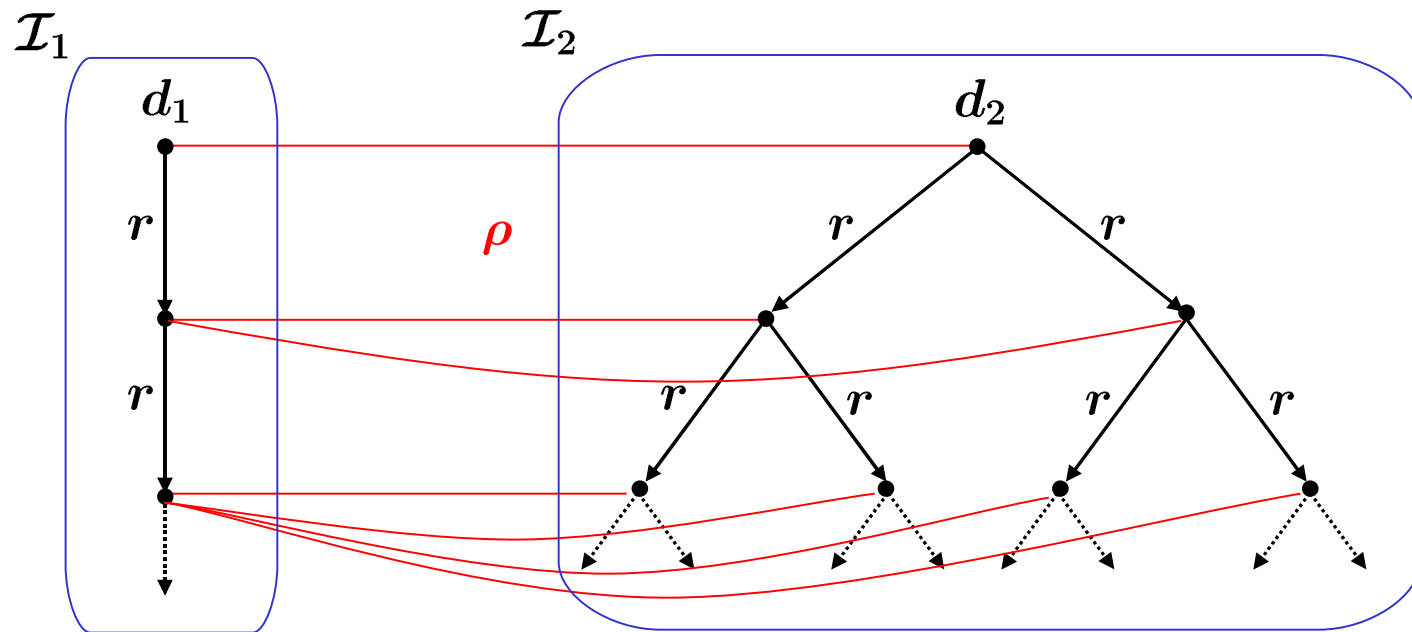
Relation  $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$  is **bisimulation** between interpretations  $\mathcal{I}_1$  and  $\mathcal{I}_2$

if  $d \rho d'$  implies that

- $d$  and  $d'$  satisfy same concept names
- each successor of  $d$  has  $\rho$ -related counterpart at  $d'$
- each successor of  $d'$  has  $\rho$ -related counterpart at  $d$



# Expressive Power of $\mathcal{ALC}$



$(\mathcal{I}_1, d_1) \sim (\mathcal{I}_2, d_2)$ : there is a bisimulation  $\rho$  between  $\mathcal{I}_1$  and  $\mathcal{I}_2$   
with  $d_1 \rho d_2$

**Lemma.**  $\mathcal{ALC}$  is **invariant under bisimulations**, i.e.,

If  $(\mathcal{I}_1, d_1) \sim (\mathcal{I}_2, d_2)$ , then  $d_1 \in C^{\mathcal{I}_1}$  iff  $d_2 \in C^{\mathcal{I}_2}$   
for all  $\mathcal{ALC}$ -concepts  $C$ .

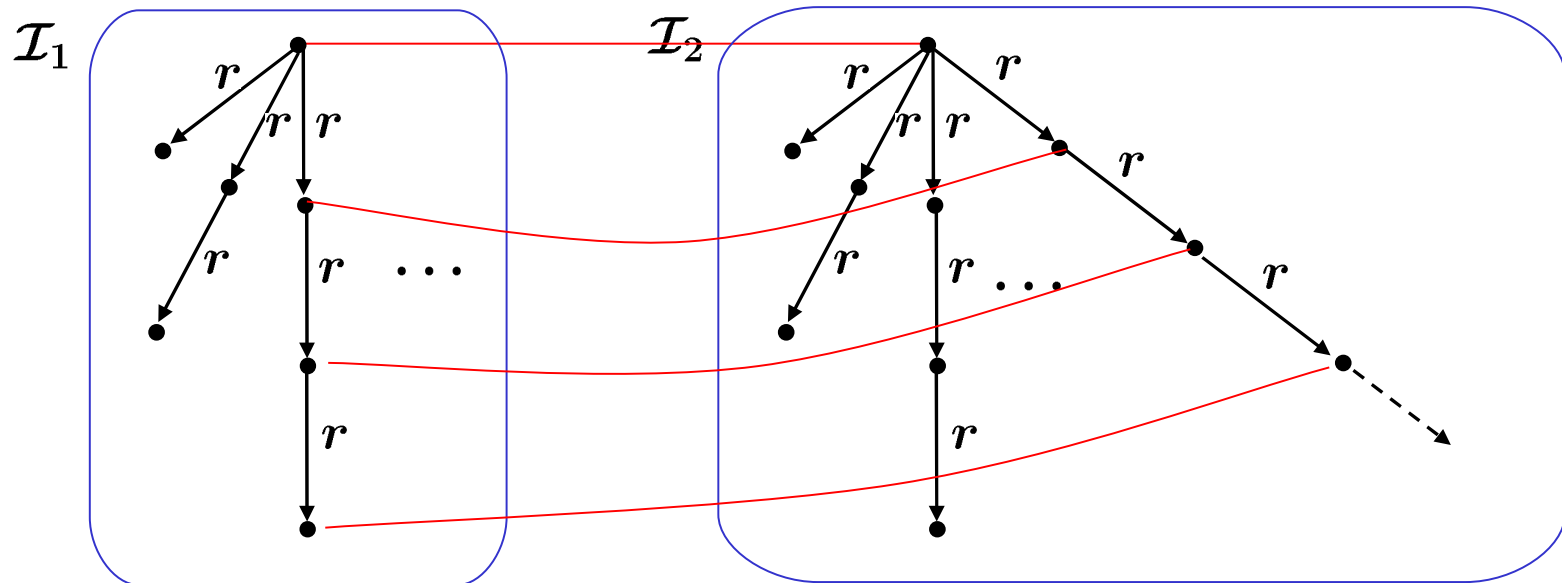
Together with example from previous slide:

$\mathcal{ALC}$  lacks expressive power for counting successors!



## Expressive Power of $\mathcal{ALC}$

The converse is **false in general**:



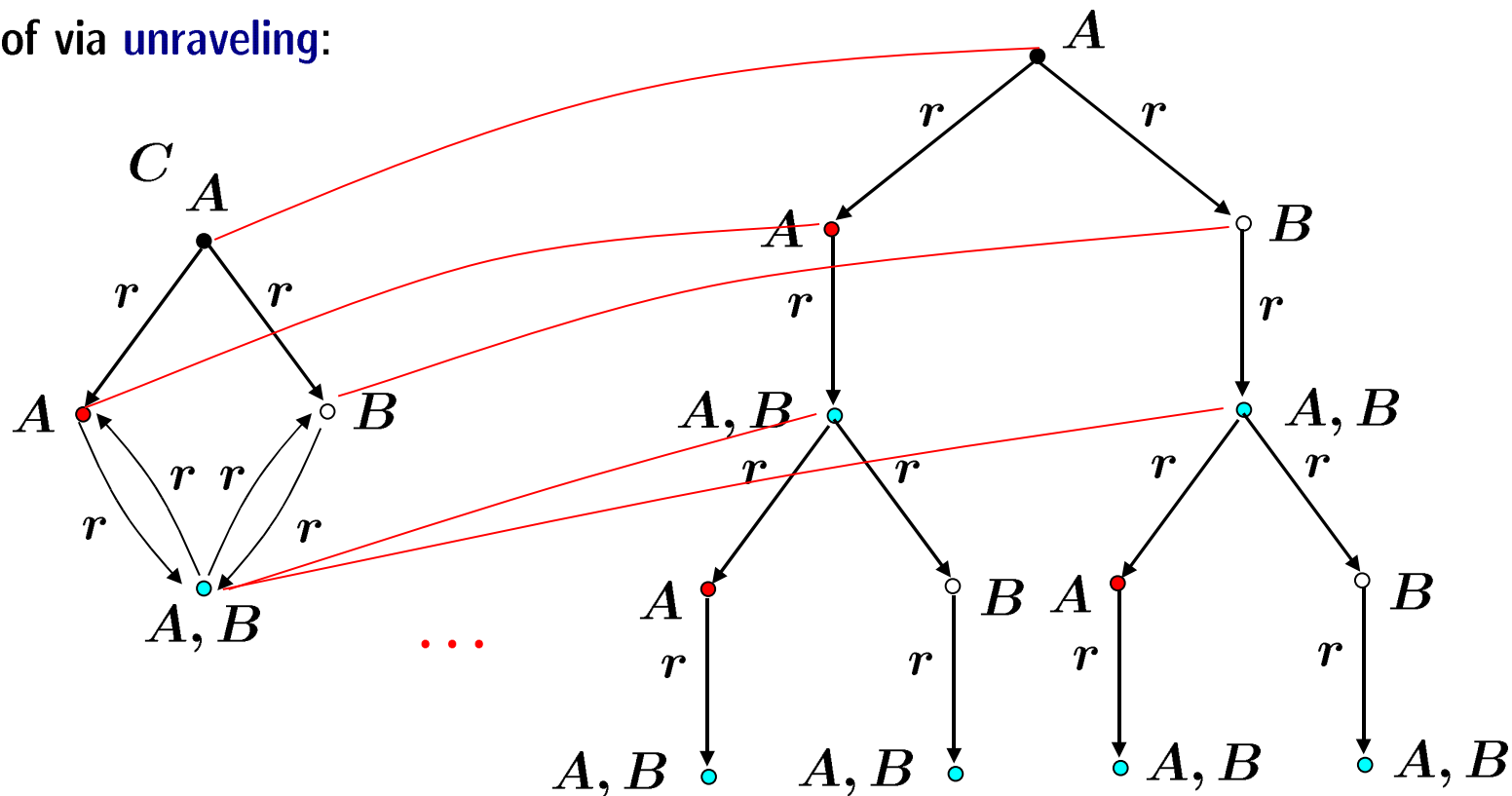
**Theorem.** An FO-formula  $\varphi$  with one free variable is equivalent to an  $\mathcal{ALC}$ -concept iff it is **invariant under bisimulation**. [vanBenthem76]



## Tree Model Property

**Theorem.** If an  $\mathcal{ALC}$ -concept  $C$  is satisfiable w.r.t. an  $\mathcal{ALC}$ -TBox  $\mathcal{T}$ , then there is a **tree-shaped model** of  $C$  and  $\mathcal{T}$

Proof via **unraveling**:



## Decidability of $\mathcal{ALC}$

Benefits of tree model property:

- tree models **computationally much simpler** than graph models  
recall, e.g., Rabin's theorem
- there are **powerful tools** for logics on trees (e.g. automata, games)

**Theorem.** In  $\mathcal{ALC}$ , satisfiability (and subsumption) is ExpTime-complete.

Many kinds of algorithms, e.g. based on:

- tree automata** (ExpTime upper bound, best case exponential)
- tableau calculus** (no ExpTime upper bound, used by most reasoners)
- Pratt-style **type elimination** (ExpTime upper bound, conceptually simple)



## Lower Bound

ExpTime-hardness: reduce word problem of **alternating Turing machines**  
whose tape is bounded polynomially [FischerLadner79]

Central ideas:

- ATMs **generalize** non-deterministic TMs:  
linear TM computations generalized to ATM **computation trees**
- alternating PSpace = ExpTime
- polysize tape can be represented using a single domain element  
(concept names such as  $A_{a,i}$ ,  $A_{h,i}$ ,  $A_q$ )
- $\mathcal{ALC}$  tree models can represent ATM computation trees





From an application perspective, the expressive power of  $\mathcal{ALC}$  is limited

OWL enriches  $\mathcal{ALC}$  in many ways, including:

- concepts  $(\leq 1\ r)$  expressing **local functionality** of roles

e.g.  $\text{Disease} \sqcap \exists \text{hasCause}.\text{Infection} \sqcap (\leq 1\ \text{hasCause})$

formal semantics:  $\forall y, y'. (r(x, y) \wedge r(x, y') \rightarrow y = y')$

- concepts  $(\leq 1\ r^-)$  for the **converse** of roles

- **nominals**, a new sort that identifies a unique domain element

e.g.  $\text{Pope}$ ,  $\text{SoccerWorldChampion}$ , but possibly also  $\text{Red}$ ,  $\text{Blue}$

Call the resulting description logic **OWL1 Core** (DL Name:  $\mathcal{ALCFIO}$ )



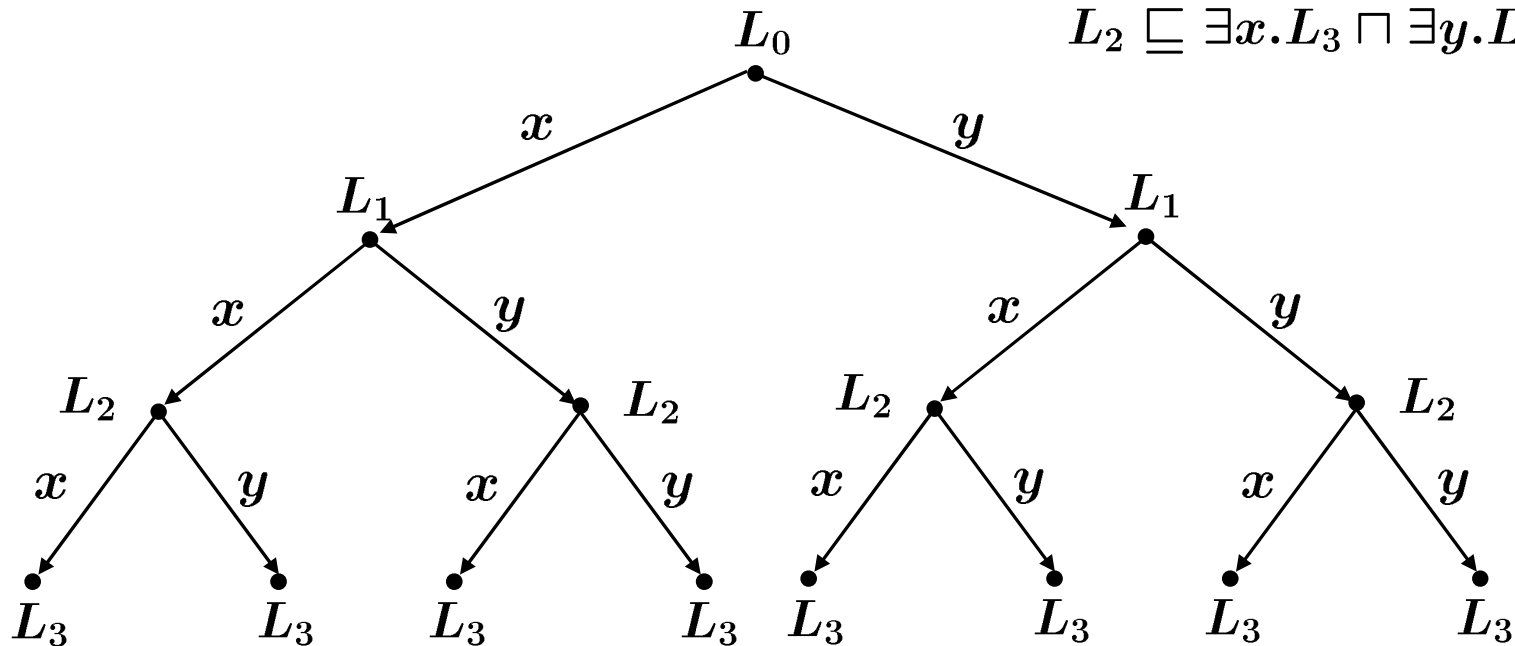
In OWL1 Core, the tree model property is **lost rather dramatically**:

- already in  $\mathcal{ALC}$ , we can easily generate a tree

$$L_0 \sqsubseteq \exists x.L_1 \sqcap \exists y.L_1$$

$$L_1 \sqsubseteq \exists x.L_2 \sqcap \exists y.L_2$$

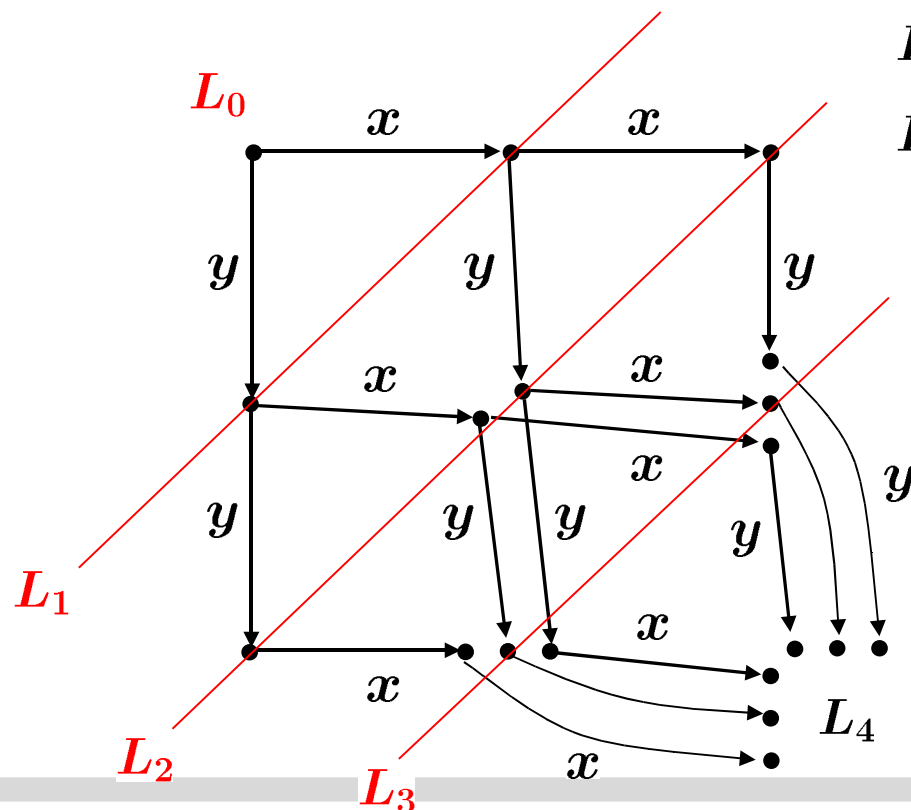
$$L_2 \sqsubseteq \exists x.L_3 \sqcap \exists y.L_3$$



## From $\mathcal{ALC}$ to OWL

In OWL1 Core, the tree model property is **lost rather dramatically**:

- already in  $\mathcal{ALC}$ , we can easily generate a tree
- now make  $L_4$  a nominal



$$L_0 \sqsubseteq \exists x.L_1 \sqcap \exists y.L_1$$

$$L_1 \sqsubseteq \exists x.L_2 \sqcap \exists y.L_2$$

$$L_2 \sqsubseteq \exists x.L_3 \sqcap \exists y.L_3$$



## From $\mathcal{ALC}$ to OWL

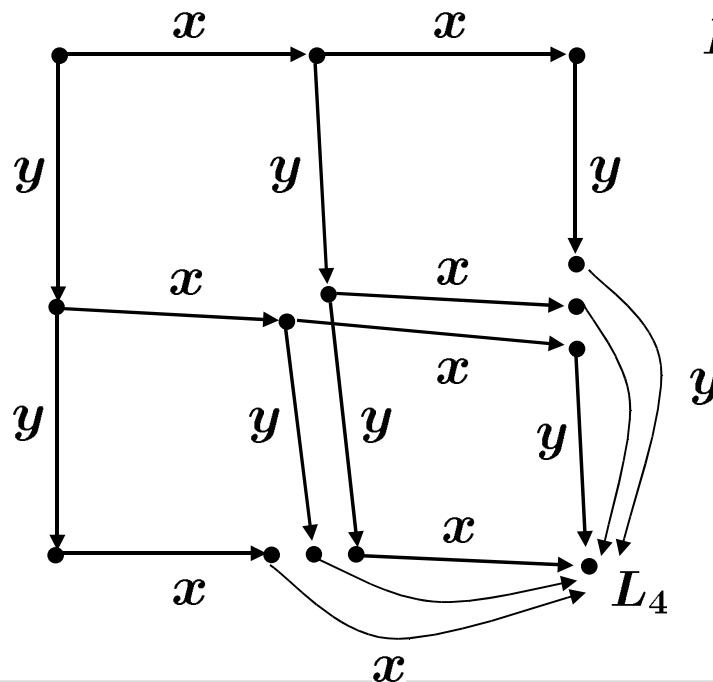
In OWL1 Core, the tree model property is **lost rather dramatically**:

- already in  $\mathcal{ALC}$ , we can easily generate a tree
- now make  $L_4$  a nominal
- make the converses of  $x$  and  $y$  functional

$$L_0 \sqsubseteq \exists x.L_1 \sqcap \exists y.L_1$$

$$L_1 \sqsubseteq \exists x.L_2 \sqcap \exists y.L_2$$

$$L_2 \sqsubseteq \exists x.L_3 \sqcap \exists y.L_3$$



## From $\mathcal{ALC}$ to OWL

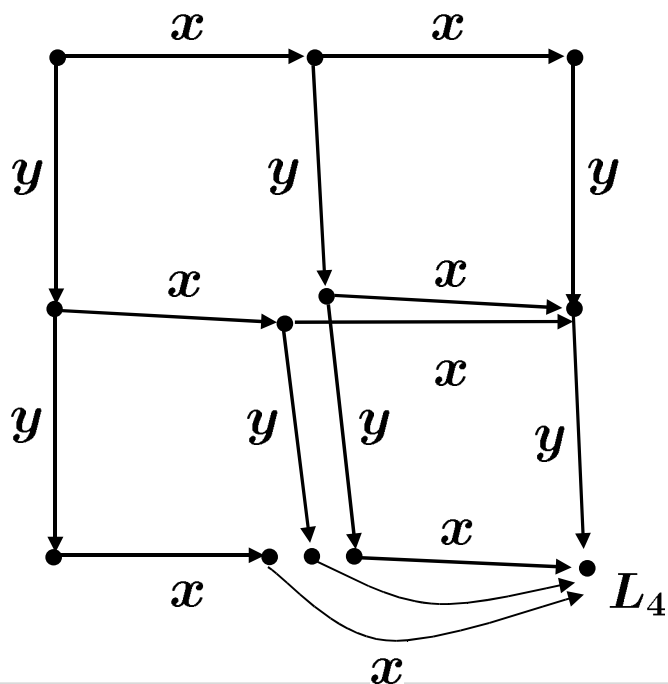
In OWL1 Core, the tree model property is **lost rather dramatically**:

- already in  $\mathcal{ALC}$ , we can easily generate a tree
- now make  $L_4$  a nominal
- make the converses of  $x$  and  $y$  functional

$$L_0 \sqsubseteq \exists x.L_1 \sqcap \exists y.L_1$$

$$L_1 \sqsubseteq \exists x.L_2 \sqcap \exists y.L_2$$

$$L_2 \sqsubseteq \exists x.L_3 \sqcap \exists y.L_3$$



## From $\mathcal{ALC}$ to OWL

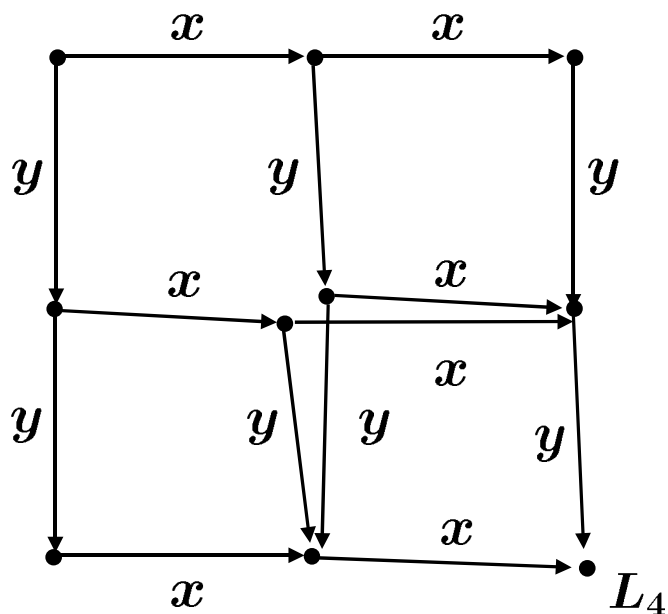
In OWL1 Core, the tree model property is **lost rather dramatically**:

- already in  $\mathcal{ALC}$ , we can easily generate a tree
- now make  $L_4$  a nominal
- make the converses of  $x$  and  $y$  functional

$$L_0 \sqsubseteq \exists x.L_1 \sqcap \exists y.L_1$$

$$L_1 \sqsubseteq \exists x.L_2 \sqcap \exists y.L_2$$

$$L_2 \sqsubseteq \exists x.L_3 \sqcap \exists y.L_3$$



## From $\mathcal{ALC}$ to OWL

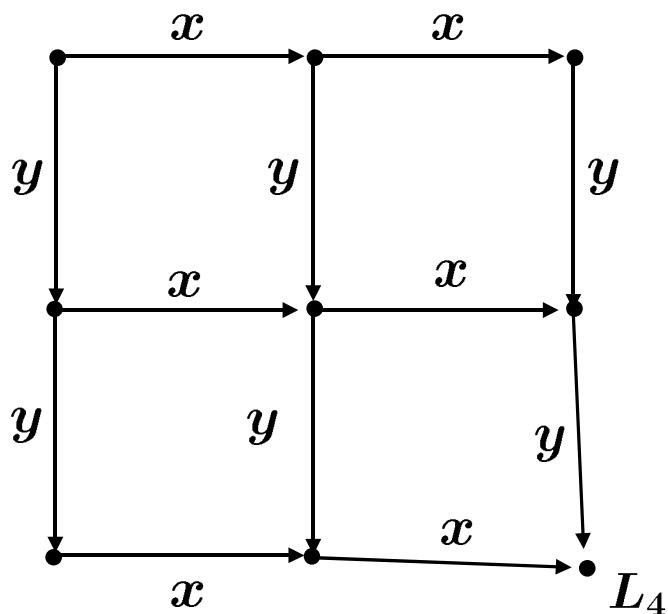
In OWL1 Core, the tree model property is **lost rather dramatically**:

- already in  $\mathcal{ALC}$ , we can easily generate a tree
- now make  $L_4$  a nominal
- make the converses of  $x$  and  $y$  functional

$$L_0 \sqsubseteq \exists x.L_1 \sqcap \exists y.L_1$$

$$L_1 \sqsubseteq \exists x.L_2 \sqcap \exists y.L_2$$

$$L_2 \sqsubseteq \exists x.L_3 \sqcap \exists y.L_3$$



### Consequences:

- the **tree model property is lost** in a rather dramatic way
- grids can represent computations of non-deterministic Turing machines
- with a small trick, we can generate a grid of **exponential size**  
(count levels in binary, not in unary)
- it follows that OWL1Core is NExpTime-hard, in fact **NExpTime-complete**  
[Tobies99]

In OWL2, we can even enforce **grids of 2-exponential size**

$\Rightarrow$  **2NExpTime-completeness** [Kazakov08]





- OWL1 and OWL2 are rather expressive  
close to, and sometimes beyond the 2-variable fragment of FO
- OWL1 and OWL2 are computationally very costly (worst case!)
- with the transition

$$\mathcal{ALC} \rightarrow \mathcal{SHIQ} \rightarrow \text{OWL1} \rightarrow \text{OWL2}$$

the promise of efficiency on natural inputs got increasingly untrue

- there are applications and reasoning tasks where this is unacceptable



## Lightweight Description Logics



## A Bit of History

Stone age of description logics (until mid-1990ies):

“We have to offer efficient reasoning and thus cannot include all Booleans”

“Every application needs at least conjunction and universal restriction”

(and thus reasoning is co-NP-complete)

The *SHIQ* era (since mid-1990ies until ??):

“ExpTime DLs can be implemented efficiently” (FaCT system by Horrocks)

“We do need the Booleans and much, much more (but want to stay decidable)!”



## A Bit of History

The  $\mathcal{EL}$  and DL-Lite era (since  $\approx 2005$ ):

“Applications need existential restrictions rather than universal ones”

“Lightweight DLs are sufficient for many applications and can be scalable”



Dominating constructors in many large-scale ontologies:

conjunction and existential restrictions

$$\text{Pericardium} \sqsubseteq \text{Tissue} \sqcap \exists \text{partOf.Heart}$$
$$\text{Pericarditis} \doteq \text{Inflammation} \sqcap \exists \text{location.Pericardium}$$
$$\text{Inflammation} \sqsubseteq \text{Disease} \sqcap \exists \text{actsOn.Tissue}$$
$$\text{Tissue} \sqcap \text{Disease} \sqsubseteq \perp$$

Large-scale ontologies usually require a highly abstract conceptual modeling



Concept language of  $\mathcal{EL}$  is “half of  $\mathcal{ALC}$ ”:

$$C ::= A \mid \top \mid \perp \mid C \sqcap D \mid \exists r.C$$

Most prominent  $\mathcal{EL}$ -ontology: **SNOMED CT**

- large scale, professionally developed **medical ontology** ( $\sim 400.000$  concepts)
- used to **systematize health care terminology**, standard e.g. in US, Canada, etc.

Satisfiability and subsumption still interreducible:

- $C$  satisfiable w.r.t.  $\mathcal{T}$  iff  $\mathcal{T} \not\models C \sqsubseteq \perp$
- $\mathcal{T} \models C \sqsubseteq D$  iff  $C \sqcap A$  unsatisfiable w.r.t.  $\mathcal{T} \cup \{C \sqcap A \sqcap D \sqsubseteq \perp\}$



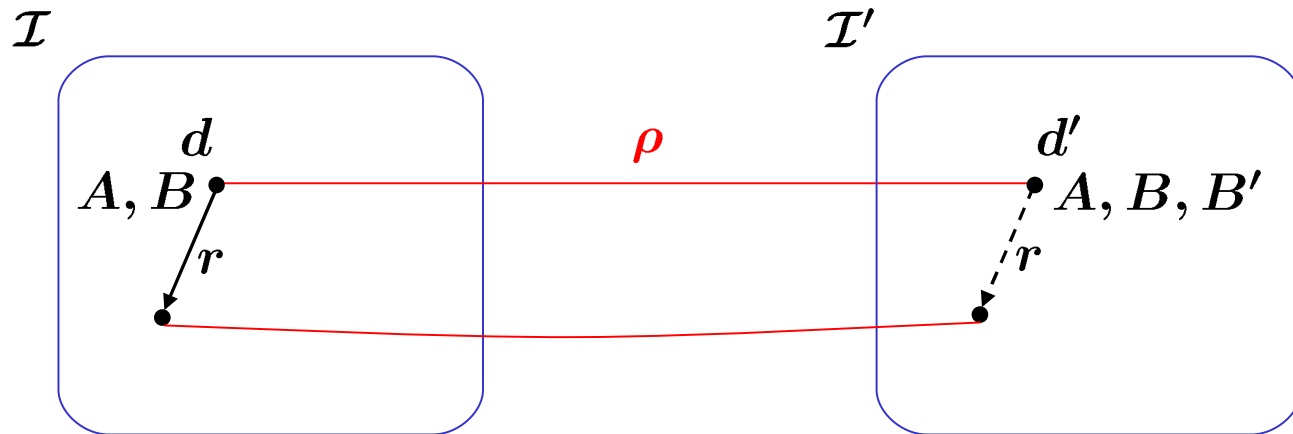
## Expressive Power of $\mathcal{EL}$

Central notion for understanding expressive power of  $\mathcal{EL}$ :

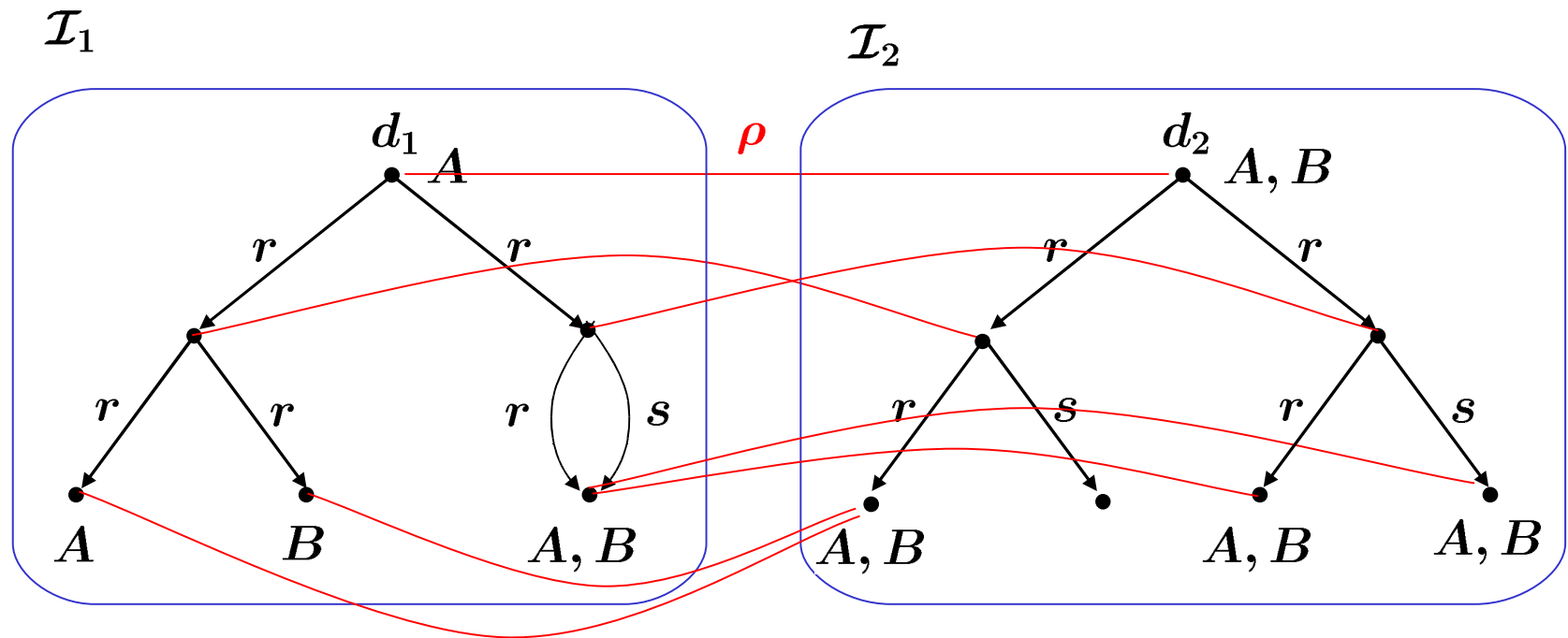
Relation  $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$  is **simulation** from interpretation  $\mathcal{I}_1$  to  $\mathcal{I}_2$

if  $d \rho d'$  implies that

- $d'$  satisfies all concept names that  $d$  satisfies
- each successor of  $d$  has  $\rho$ -related counterpart at  $d'$
- nothing else



# Expressive Power of $\mathcal{EL}$



$(\mathcal{I}_1, d_1) \preceq (\mathcal{I}_2, d_2)$ : there is a simulation  $\rho$  from  $\mathcal{I}_1$  to  $\mathcal{I}_2$   
with  $d_1 \rho d_2$



## Expressive Power of $\mathcal{EL}$

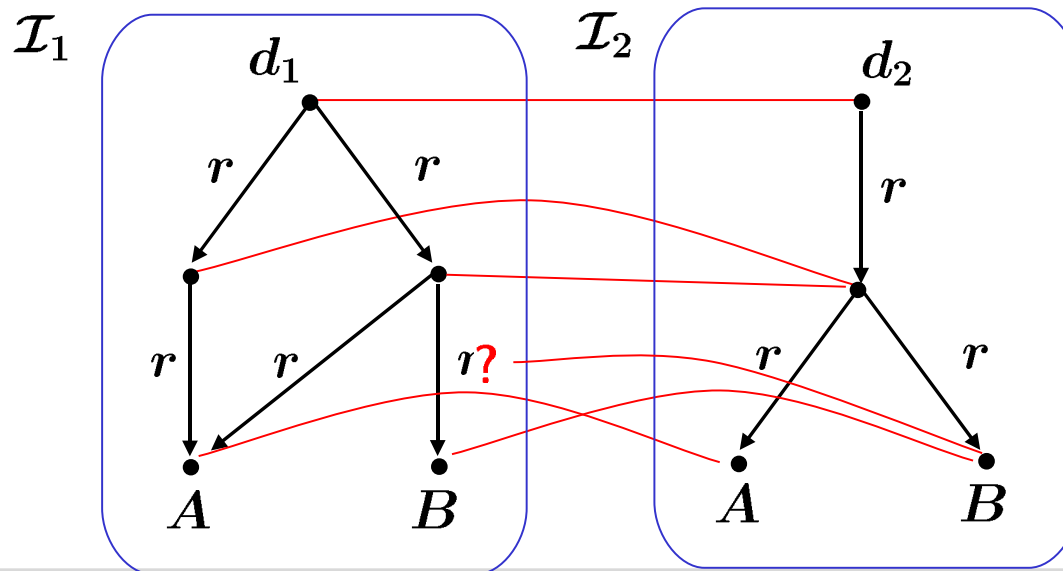
**Lemma.**  $\mathcal{EL}$  is preserved under simulations, i.e.,

if  $(\mathcal{I}_1, d_1) \preceq (\mathcal{I}_2, d_2)$ , then  $d_1 \in C^{\mathcal{I}_1}$  implies  $d_2 \in C^{\mathcal{I}_2}$

for all  $\mathcal{EL}$ -concepts  $C$ .

Thus  $\mathcal{EL}$  cannot distinguish  $(\mathcal{I}_1, d_1)$  from  $(\mathcal{I}_2, d_2)$  if they **mutually simulate**

This is not the same as bisimulation:



## Canonical Models

Since  $\mathcal{EL}$  is a fragment of  $\mathcal{ALC}$ :  $\mathcal{EL}$  has tree model property

But  $\mathcal{EL}$  satisfies a much stronger property: it has canonical tree models

**Theorem.** If an  $\mathcal{EL}$ -concept  $C$  is satisfiable w.r.t. an  $\mathcal{EL}$ -TBox  $\mathcal{T}$ ,  
then there is a tree-shaped model  $(\mathcal{M}, d)$  of  $C$  and  $\mathcal{T}$   
such that for all models  $\mathcal{I}$  of  $\mathcal{T}$  and all  $e \in C^{\mathcal{I}}$ :  $(\mathcal{M}, d) \lesssim (\mathcal{I}, e)$

Intuition: the canonical model can be found in any other model  
(in terms of a simulation)



# Canonical Models

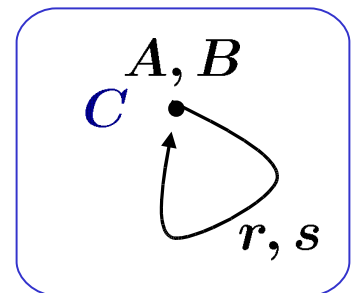
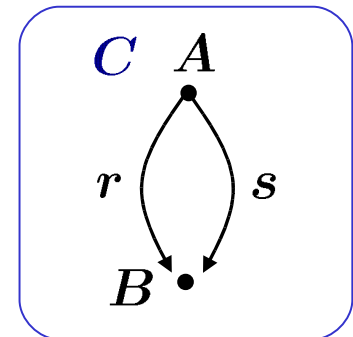
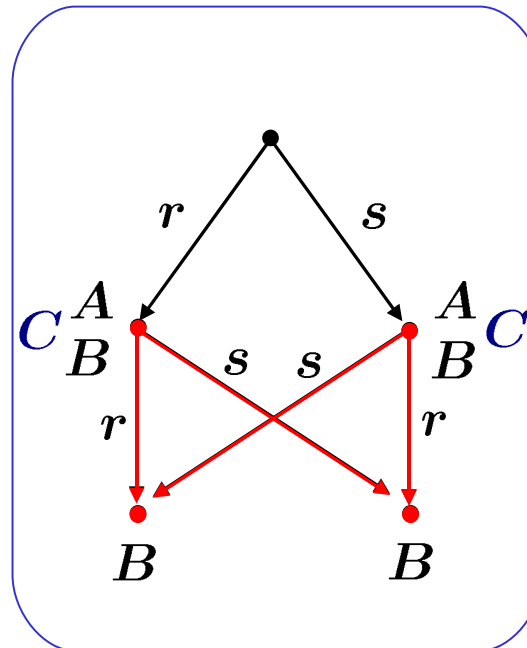
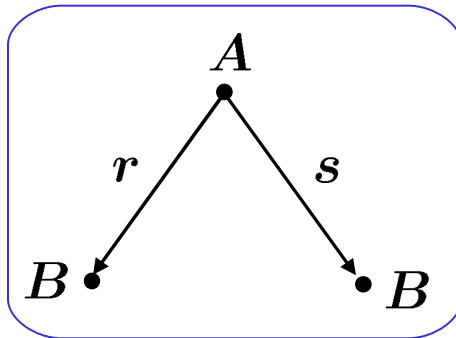
As an example, take

$$C = A \sqcap \exists r.B$$

$$\mathcal{T} = \{A \sqsubseteq \exists s.B\}$$

Models of  $\mathcal{T}$  e.g.:

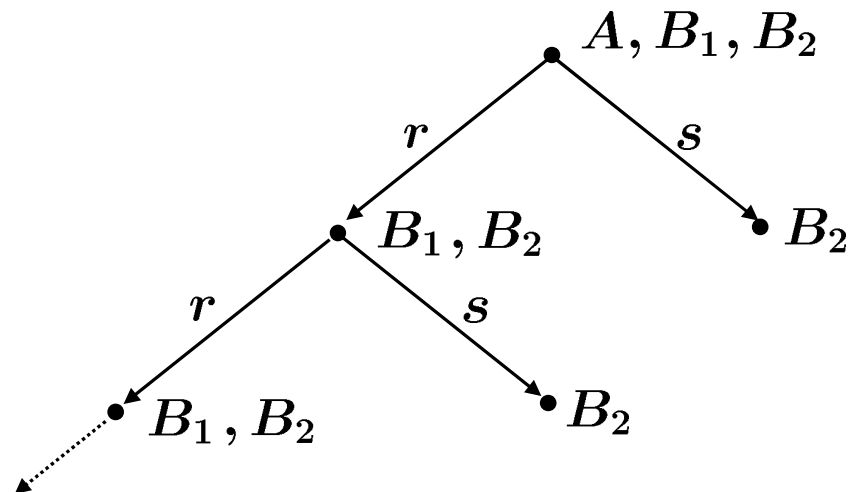
Canonical model:



## Canonical Models

Canonical models can be constructed in a straightforward way:

$$\begin{array}{lll} A \sqsubseteq B_1 & B_1 \sqsubseteq \exists r.B_1 & \exists r.B_1 \sqsubseteq B_2 \\ & B_1 \sqcap B_2 \sqsubseteq \exists s.B_2 & \end{array}$$



- This is a (tree) model of  $A$  and  $\mathcal{T}$
- Everything we have generated must be present in **every model** of  $A$  and  $\mathcal{T}$ !



Due to  $\perp$ , the canonical model construction **can fail**

and that happens exactly **when  $C$  is unsatisfiable w.r.t.  $\mathcal{T}$** :

- If we **derive  $\perp$** , then  $\perp$  is a logical consequence of  $C$  and  $\mathcal{T}$   
thus  $C$  is unsatisfiable w.r.t.  $\mathcal{T}$
- If we do **not derive  $\perp$** , then  $\mathcal{M}$  is a model of  $C$  and  $\mathcal{T}$   
thus  $C$  is satisfiable w.r.t.  $\mathcal{T}$

This is the basis for a satisfiability algorithm in  $\mathcal{EL}$ .



**Theorem.** In  $\mathcal{EL}$ , satisfiability (and subsumption) are in PTime.

[BaaderBrandtL\_\_05]

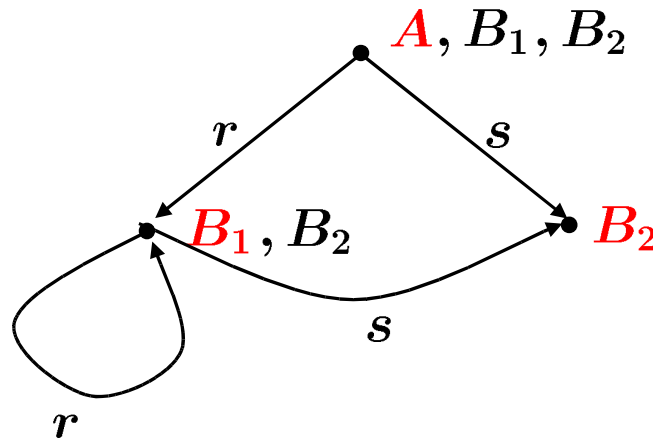
Proof approach:

- We cannot construct the infinite tree-shaped model  $\mathcal{M}$
- Instead use a **compact version** of the canonical model  $\mathcal{M}_c$

## $\mathcal{EL}$ Satisfiability

Canonical models can be constructed in a straightforward way:

$$\begin{array}{lll} A \sqsubseteq B_1 & B_1 \sqsubseteq \exists r.B_1 & \exists r.B_1 \sqsubseteq B_2 \\ & B_1 \sqcap B_2 \sqsubseteq \exists s.B_2 & \end{array}$$



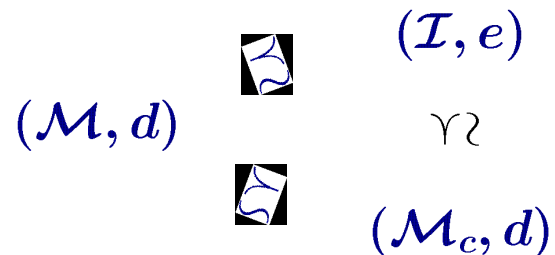
- The **unraveling** of  $\mathcal{M}_c$  is exactly  $\mathcal{M}$   
 $\implies$  construction of  $\mathcal{M}_c$  fails iff construction of  $\mathcal{M}$  fails
- $\mathcal{M}_c$  is of polynomial size, can be constructed in polynomial time



## Additional Remarks

Some **additional virtues** of  $\mathcal{M}_c$

- $\mathcal{M}_c$  is a **model of  $C$  and  $\mathcal{T}$** , too.
- just like  $\mathcal{M}$ ,  $\mathcal{M}_c$  **simulates every model of  $C$  and  $\mathcal{T}$** :



**Theorem.** An FO-formula  $\varphi$  with one free variable is equivalent to an  $\mathcal{EL}$ -concept iff it is **preserved under simulation** and has a **canonical model**.

[PiroL\_\_Wolter10]





## Extensions of $\mathcal{EL}$

PTime upper bound **can be generalized** to  $\mathcal{EL}^{++}$ , i.e.,  $\mathcal{EL}$  extended with

- **range restrictions** on roles, i.e.,  $\top \sqsubseteq \forall r.C$
  - ( ■ **domain restrictions** on roles, i.e.,  $\top \sqsubseteq \forall r^-.C$  )
  - **role implications**, i.e., TBox statements  $r_1 \circ \dots \circ r_n \sqsubseteq r$
  - ...
- } OWL EL Profile

Other extensions cause a **jump back to ExpTime**, e.g.

- **disjunctions**  $C \sqcup D$
- **universal restrictions**  $\forall r.C$
- **number restrictions**  $(\geq 2 r)$

Interesting: **no extension between PTime and ExpTime known** (dichotomy?)



**Theorem.** In  $\mathcal{EL} + \sqcup$ , satisfiability (and subsumption) are ExpTime-complete.  
[BaaderBrandtL\_\_05]

Proof: **reduction from satisfiability** of concept name  $A_0$  w.r.t.  $\mathcal{ALC}$ -TBox  $\mathcal{T}$

**Step 1:** Replace universal restrictions in  $\mathcal{T}$  with existential ones:

$$\forall r.C \quad \text{becomes} \quad \neg \exists r. \neg C$$

**Step 2:** Modify  $\mathcal{T}$  so that negation is applied only to concept **names**

$$A \sqsubseteq \exists s.(B' \sqcup \neg \exists r.B) \quad \text{becomes} \quad A \sqsubseteq \exists s.(B' \sqcup \neg X)$$

$$X \doteq \exists r.B$$

( $X$  a fresh concept name)



**Theorem.** In  $\mathcal{EL} + \sqcup$ , satisfiability (and subsumption) are ExpTime-complete.  
[BaaderBrandtL\_\_05]

Proof: **reduction from satisfiability** of concept name  $A_0$  w.r.t.  $\mathcal{ALC}$ -TBox  $\mathcal{T}$

**Step 3:** Remove negation entirely from  $\mathcal{T}$

- Replace each  $\neg X$  with  $\overline{X}$ ,  $\overline{X}$  a fresh concept name
- Ensure **correct behaviour** of  $\overline{X}$ :

$$\begin{aligned}\top &\sqsubseteq X \sqcup \overline{X} \\ X \cap \overline{X} &\sqsubseteq \perp\end{aligned}$$

Resulting TBox  $\mathcal{T}'$  is in  $\mathcal{EL} + \sqcup$  and  $A_0$  sat w.r.t.  $\mathcal{T}$  iff  $A_0$  sat w.r.t.  $\mathcal{T}'$



**Theorem.** In  $\mathcal{EL} + \forall r.C$  and  $\mathcal{EL} + (\geq 2 r)$ , satisfiability is ExpTime-complete.

[BaaderBrandtL\_\_05]

Proof: **reduction from satisfiability** of concept name  $A_0$  w.r.t.  $\mathcal{EL} + \sqcup\text{-TBox } \mathcal{T}$

We can assume that disjunction occurs only in the form

$$\begin{array}{ccc}
 A_1 \sqcup A_2 \sqsubseteq A & \text{and} & A \sqsubseteq B_1 \sqcup B_2 \\
 \underbrace{\hspace{1.5cm}} & & \mid \\
 = A_1 \sqsubseteq A, A_2 \sqsubseteq A & & \text{replace by} \\
 & & A \sqcap \exists r.T \sqsubseteq B_1 \\
 & & A \sqcap \forall r.X \sqsubseteq B_2 \quad r, X \text{ fresh}
 \end{array}$$



**Theorem.** In  $\mathcal{EL} + \forall r.C$  and  $\mathcal{EL} + (\geq 2 r)$ , satisfiability is ExpTime-complete.

[BaaderBrandtL\_\_05]

Proof: **reduction from satisfiability** of concept name  $A_0$  w.r.t.  $\mathcal{EL} + \sqcup\text{-TBox } \mathcal{T}$

We can assume that disjunction occurs only in the form

$$\underbrace{A_1 \sqcup A_2 \sqsubseteq A}_{= A_1 \sqsubseteq A, A_2 \sqsubseteq A} \quad \text{and} \quad A \sqsubseteq B_1 \sqcup B_2$$

replace by

$$A \sqsubseteq \exists r.X \sqcap \exists r.Y$$

$$A \sqcap \exists r.(X \sqcap Y) \sqsubseteq B_1$$

$r, X, Y$  fresh

$$A \sqcap (\geq 2 r) \sqsubseteq B_2$$



## Extensions of $\mathcal{EL}$

Call an extension of  $\mathcal{EL}$  **convex** if:

$$\mathcal{T} \models C \sqsubseteq D_1 \sqcup D_2 \quad \text{implies} \quad \mathcal{T} \models C \sqsubseteq D_i \text{ for some } i \in \{1, 2\}$$

$\mathcal{EL} + \forall r.C$  is not convex:

$$\emptyset \models \top \sqsubseteq \exists r.\top \sqcup \forall r.X, \text{ but } \emptyset \not\models \top \sqsubseteq \exists r.\top \text{ and } \emptyset \not\models \top \sqsubseteq \forall r.X$$

The reductions show: if an extension of  $\mathcal{EL}$  is not convex, it is ExpTime-hard.

Interestingly, the converse does not hold!

Easy to prove:

Existence of canonical models  $\mathcal{M}$  implies convexity:



## Extensions of $\mathcal{EL}$

Consider  $\mathcal{EL}$  extended with **inverse existential restrictions**:

$\exists r^-.C$  has semantics  $\{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}} : (e, d) \in r^{\mathcal{I}}\}$

**Theorem.**  $\mathcal{EL} + \exists r^-.C$  is convex, but satisfiability is ExpTime-complete.

[BaaderBrandtL\_05]

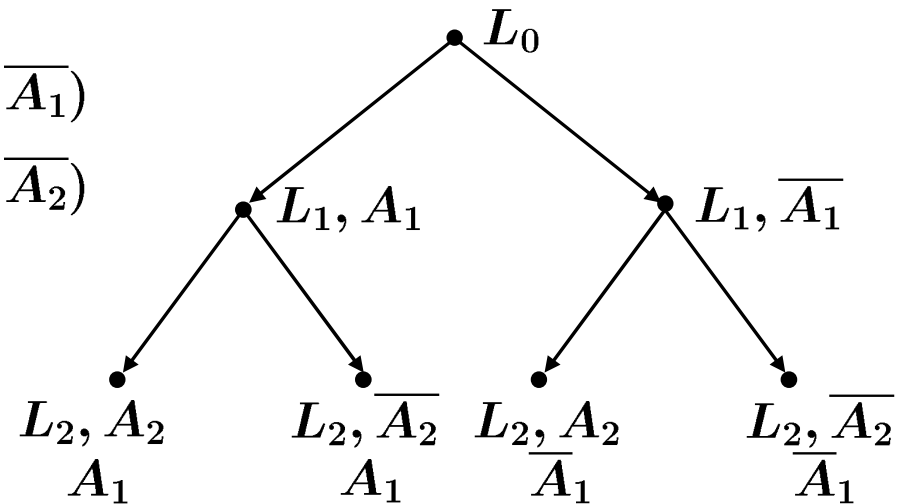
Here only: canonical models can become **exponentially large**

$$L_0 \sqsubseteq \exists r.(L_1 \sqcap A_1) \sqcap \exists r.(L_1 \sqcap \overline{A_1})$$

$$L_1 \sqsubseteq \exists r.(L_2 \sqcap A_2) \sqcap \exists r.(L_2 \sqcap \overline{A_2})$$

$$L_2 \sqcap \exists r^-.A_1 \sqsubseteq A_1$$

$$L_2 \sqcap \exists r^-. \overline{A_1} \sqsubseteq \overline{A_1}$$



Merging leaves destroys canonicity!



## Discussion

- $\mathcal{EL}$  is a natural ontology language for a high level of abstraction
- satisfiability and subsumption can be computed in polytime
- this has led to standardization as OWL EL profile of OWL2
- efficient reasoners are available, e.g. CEL (Dresden), SnoRocket (Brisbane)  
based on canonical models, very robust, classify SNOMED CT in <10min
- algorithms have been generalized to Horn- $\mathcal{SHIQ}$   
reasoner CB (Oxford)





The historic choice of **universal restrictions** instead of **existential restrictions** leads to much worse computational behaviour

Complexity of **subsumption** in  $\mathcal{FL}_0$ , constructors  $\top$ ,  $(\perp)$ ,  $\sqcap$ ,  $\forall r.C$ :

- empty TBox: tractable [BrachmanLevesque84]
- acyclic TBox: co-NP-complete [Nebel90]
- cyclic TBox: PSpace-complete [KazakovDeNivelle03]
- general TBox: ExpTime-complete [BaaderBrandtL\_05,Hofmann05]

## A Glimpse at $\mathcal{FL}_0$

Clearly,

$$\forall r.(A \sqcap B) \equiv \forall r.A \sqcap \forall r.B$$

Thus, every  $\mathcal{FL}_0$ -concept is equivalent to one of the form

$$\begin{aligned} & \forall r_{1,1}.\forall r_{1,2}.\dots\forall r_{1,n_1}.A_1 \\ & \sqcap \forall r_{2,1}.\forall r_{2,2}.\dots\forall r_{2,n_2}.A_2 \\ & \quad \dots \\ & \sqcap \underbrace{\forall r_{k,1}.\forall r_{k,2}.\dots\forall r_{k,n_k}}_{\text{(finite) words over the alphabet of role names}}.A_k \end{aligned}$$

(finite) words over the alphabet of role names

Grouping according to concept name achieves the following **normal form**

$$\forall L_1.A_1 \sqcap \forall L_2.A_2 \sqcap \dots \sqcap \forall L_m.A_m$$

finite formal languages over the alphabet of role names



We consider subsumption instead of satisfiability

Subsumption in  $\mathcal{FL}_0$  (without TBoxes):

$$C = \forall L_1.A_1 \sqcap \forall L_2.A_2 \sqcap \dots \sqcap \forall L_m.A_m$$

$$D = \forall M_1.A_1 \sqcap \forall M_2.A_2 \sqcap \dots \sqcap \forall M_m.A_m$$

Then  $C \sqsubseteq D$  iff  $L_i \supseteq M_i$  for  $1 \leq i \leq m$  (\*)

**Theorem.** Subsumption in  $\mathcal{FL}_0$  without TBoxes is in PTime.

Intuitively (\*) still holds with acyclic TBoxes,

but sets  $L_i$  can be described compactly, get exponentially large



Reduction from  $\overline{3SAT}$  to  $\mathcal{FL}_0$ -subsumption w.r.t. TBoxes:

Take a 3-formula

$$\varphi = (\ell_{1,1} \vee \ell_{1,2} \vee \ell_{1,3}) \wedge \cdots \wedge (\ell_{n,1} \vee \ell_{n,2} \vee \ell_{n,3})$$

over the variables  $x_1, \dots, x_k$

Ideas:

- use two role names  $t$  and  $f$  representing “true” and “false”
- represent truth assignments as words over  $\{t, f\}$  of length  $k$
- as the target subsumption  $C \sqsubseteq D$ , use

$C = \forall L_C.A$ ,  $L_C$  the set of truth assignments that make  $\varphi$  false

$D = \forall L_D.A$ ,  $L_D$  the set of all truth assignments



## A Glimpse at $\mathcal{FL}_0$

To be done: describe  $C$  and  $D$  with polynomial-size TBox:

$D$  is easy:

$$L_i \equiv \forall t.L_{i+1} \sqcap \forall f.L_{i+1} \quad \text{for } 1 \leq i \leq n$$

$$L_{n+1} \equiv A$$

$$D \equiv L_0$$

$C$  too (basically)

**Theorem.** Subsumption in  $\mathcal{FL}_0$  w.r.t. TBoxes is co-NP-hard.



## Instance Data and Query Answering



In recent years, **exciting new reasoning problems** have popped up; e.g.:

- **conjunctive query answering** over instance data  
w.r.t. a background TBox
- problems related to the **modularity of TBoxes**:
  - does a given subset  $\mathcal{T}' \subseteq \mathcal{T}$  **say everything** about  
a given signature  $\Sigma$  that  $\mathcal{T}$  does?
  - given a signature  $\Sigma$ , **extract an as-small-as-possible  
subset**  $\mathcal{T}' \subseteq \mathcal{T}$  that says the same about  $\Sigma$  as  $\mathcal{T}$
- problems related to **privacy issues**  
e.g. controlled interfaces to TBox / instance data



conservative  
extensions

Ontologies are increasingly used with **instance data**, e.g.:

**Clinical document architecture (CDA)** becomes standard medical data format

CDA medical codes based on **SNOMED CT terminology**

Ontology can be exploited for **interpreting data / deriving additional answers**

**ABox**: finite set of ground facts, e.g.:

Patient( $p$ )

finding( $p, d$ )

Pericarditis( $d$ )

Information in ABoxes is incomplete (**open world semantics**)

E.g., a patient record would not include **Inflammation( $d$ )**, though it is true.





## ABoxes

TBox allows more complete query answers

ABox

Patient( $p$ )	Inpatient( $p$ )
inWard( $p, w$ )	$\neg$ Intensive( $w$ )

TBox

Inpatient $\sqsubseteq$ $\exists$ finding.Disease
$\exists$ inWard. $\neg$ Intensive $\sqsubseteq$ $\forall$ finding. $\neg$ LiveThreatening

Then  $p$  is an answer to query

$\exists y.$ Patient( $x$ )  $\wedge$  finding( $x, y$ )  $\wedge$   $\neg$ LiveThreatening( $y$ )



More formally:

- Model of ABox  $\mathcal{A}$ : interpretation satisfying all facts in  $\mathcal{A}$
- Answers to query  $q$  for ABox  $\mathcal{A}$  w.r.t. TBox  $\mathcal{T}$ :

Certain answers, i.e., answers common to all models  $\mathcal{I}$  of  $\mathcal{A}$  and  $\mathcal{T}$

Closely related to query answering in incomplete databases

(but with a different kind of schema constraints)



### Instance queries

take the form  $C(v)$ ,  $v$  a **variable**.

technically **close to subsumption**, almost always of **same complexity**

### Conjunctive queries

take the form  $\exists \vec{v}. \varphi(\vec{v}, \vec{v}')$ , with  $\varphi$  a **conjunction of atoms**  $A(v)$  or  $r(v, v')$


$\vec{v}'$  the **answer variables**,  $\vec{v}$  the **quantified variables**

**generalize instance queries**, but are more interesting

**Select-Project-Join fragment of SQL**

### FO/SQL queries

**generalize conjunctive queries**, but: FO sentence  $\varphi$  valid iff  $\emptyset, \emptyset \models \varphi$

  
TBox      ABox



In patient databases and other large-scale applications:

- Efficiency and scalability of query answering is crucial
- Query answering in expressive DLs is computationally costly

	satisfiability	query answering
$\mathcal{ALC}$	ExpTime	ExpTime
$\mathcal{ALC} + \exists r^-.C$	ExpTime	2ExpTime
$\mathcal{SHIQ}$	ExpTime	2ExpTime
OWL1 Core	NExpTime	decidable
OWL1	NExpTime	decidability open



Most popular approach to achieve scalability:

Implement DL query answering based on relational database systems

Obvious problem: conventional RDBM unaware of TBoxes

- **Solution I:** query rewriting — “put TBox into query”
- **Solution II:** data completion — “put TBox into data”



**Solution I:** query rewriting — “put TBox into query”



The query rewriting approach: [Calvanese, deGiacomo, Lenzerini et al.05]

- ABox stored in DB system as relational instance
- CQ is rewritten to FO/SQL query to incorporate TBox
- Rewritten query executed by relational DB system

Enables use of off-the-shelf DB systems!

**Mission statement:** given CQ  $q$  and  $\mathcal{T}$ , rewrite  $q$  into FO query  $q'$  such that

$\mathcal{A}, \mathcal{T} \models q[a_1, \dots, a_n]$  iff  $db_{\mathcal{A}} \models q'[a_1, \dots, a_n]$  for all  $\mathcal{A}, a_1, \dots, a_n$ .

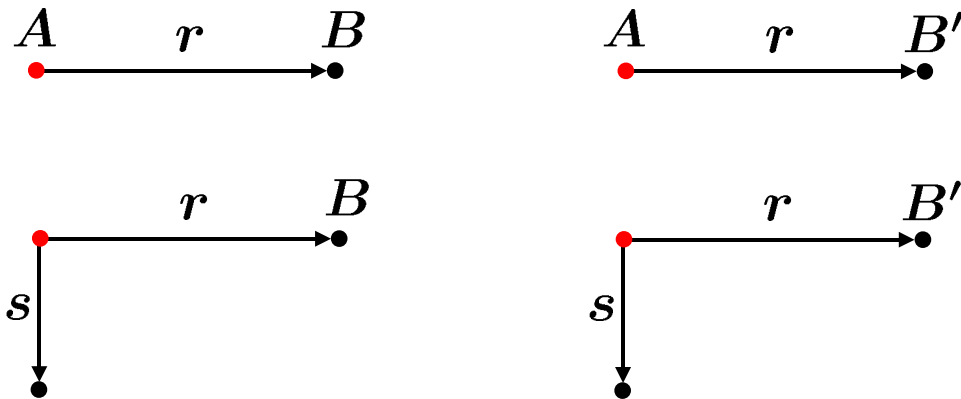


## Query Rewriting—Example 1

Query       $A \xrightarrow{r} B$        $\exists y.(A(x) \wedge r(x, y) \wedge B(y))$

TBox       $\exists s.\top \sqsubseteq A$        $B' \sqsubseteq B$

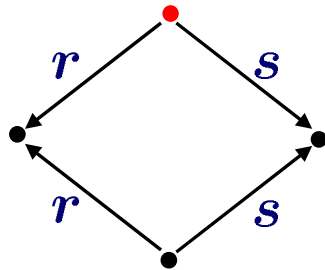
Rewritten query is **disjunction** of:





## Query Rewriting—Example 2

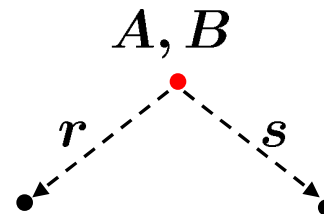
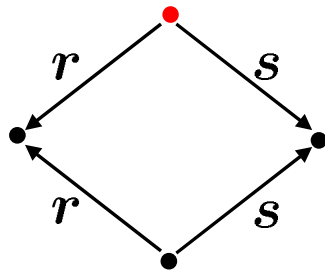
Query



TBox

$$A \sqsubseteq \exists r. \top \quad B \sqsubseteq \exists s. \top$$

Rewritten query is **disjunction** of:



For which DLs does this work?



### Data complexity:

- In DBs: measure complexity **only in size of data**, not of query
- In DLs: measure complexity **only in size of data**, neither of query **nor TBox**

**Theorem.** The query rewriting approach only works for DLs for which CQ entailment is in  $AC_0$  regarding data complexity. [Calvanese et al. 05]

### Proof:

- FO query answering is in  $AC_0$  regarding data complexity
- measured input (data) is left unchanged
- measured / non-measured inputs are **not mixed** in the rewriting



## Query Rewriting

Data complexity of  
DLs we have met:

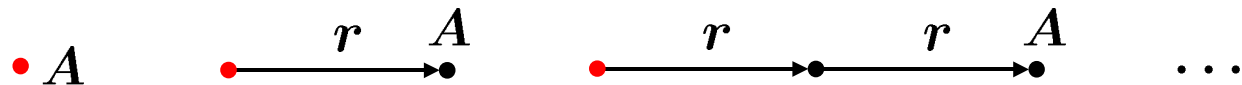
$\mathcal{EL}$	PTime-complete
$\mathcal{ALC}$ and above	NP-complete

Why query rewriting cannot be used for  $\mathcal{EL}$ :

Query             $\bullet A$              $A(x)$

TBox             $\exists r.A \sqsubseteq A$

Rewritten query is **disjunction** of:



We need  $\exists y.r^*(x, y) \wedge A(y)$ , but transitive closure not FO-expressible



## DL-Lite

**DL-Lite:** a lightweight DL with  $AC_0$  data complexity [Calvanese et al.05]

Basic version: TBox statements of the form

$$C \sqsubseteq D \quad C \sqsubseteq \neg D$$

where  $C, D$  are of the form  $A$ ,  $\exists r.\top$ , and  $\exists r^-\top$

For example:  $\text{Professor} \sqsubseteq \exists \text{teachesTo}.\top$      $\exists \text{teachesTo}^-\top \sqsubseteq \text{Student}$

$$\text{Professor} \sqsubseteq \neg \text{Student}$$

**DL-Lite:**

- inexpressive, but can encode ER diagrams and UML class diagrams
- admits the query rewriting approach
- underlies OWL QL profile of OWL2



**Solution II:** data completion — “put TBox into data”



**Limitations** of the query rewriting approach:

- Works only for  $AC_0$ -DLs, i.e., **only for DL-Lite**
- Query rewriting **blows up exponentially**  $O(|\mathcal{T}|^{|q|})$   
**performance problems** with large queries / large TBoxes

The data completion approach **avoids both problems**

in particular, it **works for  $\mathcal{EL}$ -TBoxes**



The data completion approach: [L\_\_TomanWolter08]

- Incorporate TBox **into the ABox**, not into the query
- To deal with existential restrictions and **avoid infinite databases**:  
**eagerly reuse constants, producing spurious cycles (and more)**  
(similar to compact canonical model vs. canonical tree model)
- To nevertheless obtain correct answers: use **query rewriting**

Also enables use of **off-the-shelf DB systems!**

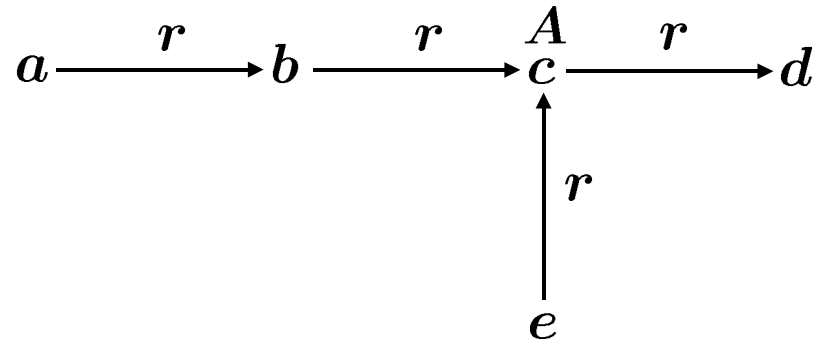


## Data Completion—Example 1

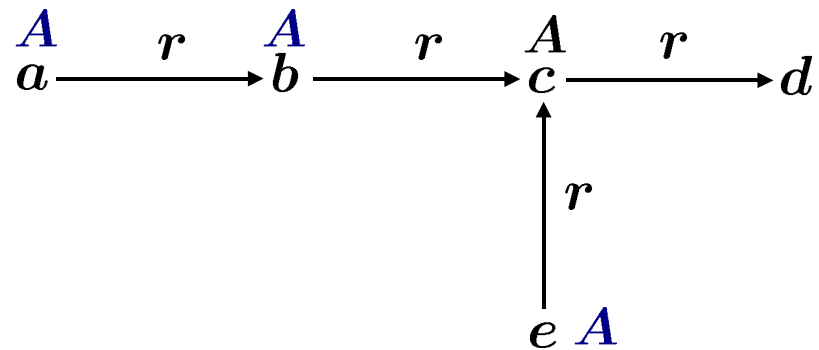
TBox

$\exists r.A \sqsubseteq A$

ABox



Completed ABox:



Query

$A(x)$

Answer

$a, b, c, e$





## Data Completion—Example 2

TBox  $A \sqsubseteq \exists s.B \quad \exists s.B \sqsubseteq A' \quad \exists r.(A \sqcap A') \sqsubseteq B$

ABox  $a \xrightarrow{r} b$

Completed ABox:

$$\begin{array}{c}
 \begin{array}{ccc}
 B & & A, A' \\
 a \xrightarrow{r} & & b \\
 & & \downarrow s \\
 & & c \quad B, \text{Ex}
 \end{array}
 \end{array}$$

Query	$B(v)$	Answer	$a, c$
-------	--------	--------	--------

Rewritten query	$B(v) \wedge \neg \text{Ex}(v)$	Answer	$a$
-----------------	---------------------------------	--------	-----



## Data Completion—Example 2

TBox

$$A \sqsubseteq \exists s.B \quad \exists s.B \sqsubseteq A' \quad \exists r.(A \sqcap A') \sqsubseteq B$$

ABox

$$a \xrightarrow{r} b \quad \begin{matrix} A \\ b \end{matrix}$$

Completed ABox:

$$\begin{array}{ccc} B & & A, A' \\ a \xrightarrow{r} & b & \\ & \downarrow s & \\ & c & B, Ex \end{array}$$

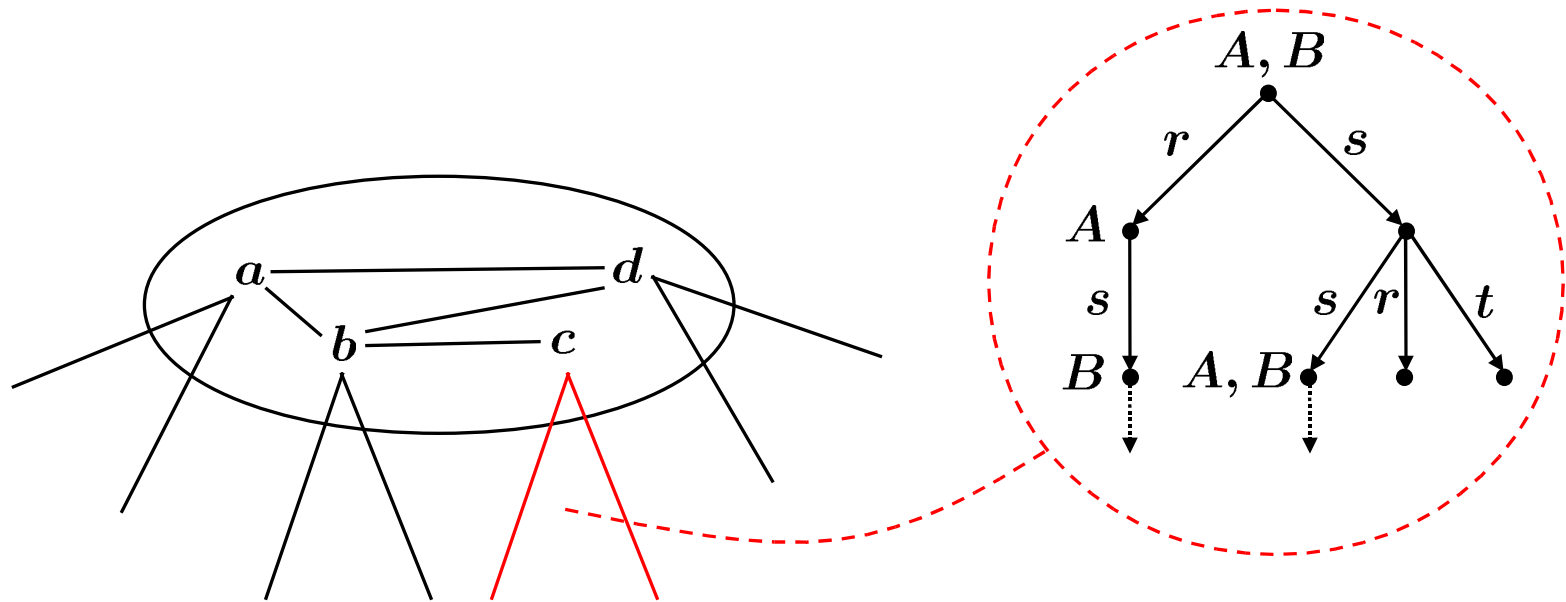
ABox completion means **building the canonical model**

(for an ABox instead of for a concept)



## Data Completion—Example 2

General shape of canonical model built for an ABox:



Problem: canonical model can get **infinite**, database can't

## Data Completion—Example 3

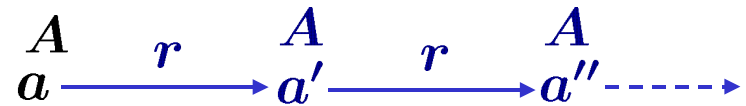
TBox

$$A \sqsubseteq \exists r.A$$

ABox

$$\frac{A}{a}$$

Completed ABox:



Database cannot be infinite.

$\Rightarrow$  build compact canonical model!

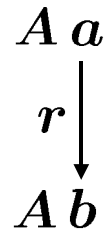


## Data Completion—Example 3

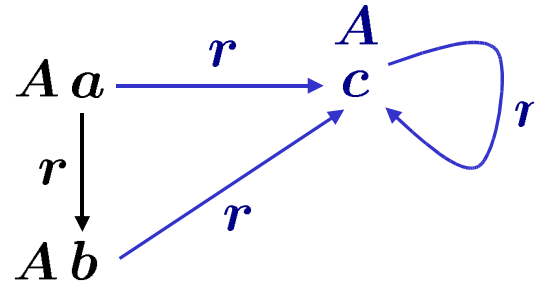
TBox

$$A \sqsubseteq \exists r.A$$

ABox



Completed ABox:



Wrong answer to some queries, e.g.

$$\exists y. r(\mathbf{x}, y) \wedge r(y, y)$$

answer  $\{a, b\}$ , should be  $\emptyset$

$$\exists y. r(\mathbf{x}, y) \wedge r(\mathbf{x}', y) \wedge r(\mathbf{x}, \mathbf{x}')$$

answer  $\{(a, b)\}$ , should be  $\emptyset$



## Data Completion

Problem:

infinite, tree-shaped canonical model  $\mathcal{M}$  gives correct answers to all queries,  
compact version  $\mathcal{M}_c$  does not

Solution:

Rewrite CQ  $q$  into FO query  $q'$  so that

answers to  $q'$  in  $\mathcal{M}_c = \text{answers to } q \text{ in } \mathcal{M}$

Implementation: add query conjuncts expressing that

- Variable on a query cycle cannot be mapped to an Ex element
- If  $r(x, y), s(x', y)$  in query and  $r \neq s$ , then  $y$  not mapped to Ex
- If  $r(x, y), r(x', y)$  in query and  $y$  mapped to Ex, then  $x = x'$

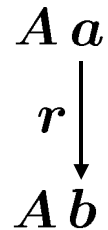


## Data Completion—Example 3

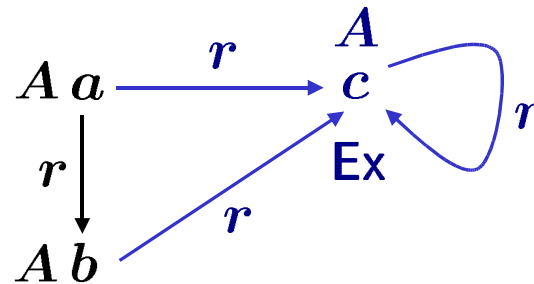
TBox

$$A \sqsubseteq \exists r.A$$

ABox



Completed ABox:



$$q = \exists y. r(\mathbf{x}, y) \wedge r(y, y)$$

answer  $\{a, b\}$

$$q' = \exists y. r(\mathbf{x}, y) \wedge r(y, y) \wedge \neg \text{Ex}(\mathbf{x}) \wedge \neg \text{Ex}(y)$$

answer  $\emptyset$

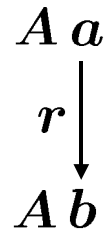


## Data Completion—Example 3

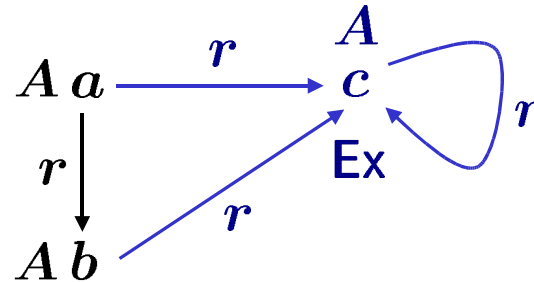
TBox

$$A \sqsubseteq \exists r.A$$

ABox



Completed ABox:



$$q = \exists y.r(\mathbf{x}, y) \wedge r(\mathbf{x}', y) \wedge r(\mathbf{x}, \mathbf{x}')$$

answer  $\{(a, b)\}$

$$q' = \exists y.r(\mathbf{x}, y) \wedge r(\mathbf{x}', y) \wedge r(\mathbf{x}, \mathbf{x}'))$$

answer  $\emptyset$

$$\wedge \neg \text{Ex}(\mathbf{x}) \wedge \neg \text{Ex}(\mathbf{x}') \wedge (\text{Ex}(y) \rightarrow \mathbf{x} = \mathbf{x}')$$





### Wrapup:

- ❶ Data completion approach works for EL and DL-Lite [KR10], results only in polynomial blowup of the query
- ❷ Requires authority over the data, blows up the data (polynomially)
- ❸ Extends to role hierarchies, domain and range restrictions (but transitive roles and general role inclusions are challenging)
- ❹ Limitation: for DLs whose data complexity is not in PTime there must be a (worst case) exponential blowup of the data





Questions?

PS: Slides are on my homepage

PPS: Somebody interested in a PhD/Postdoc position?





## Decidability of $\mathcal{ALC}$

Tree model property is a good explanation for **decidability** of  $\mathcal{ALC}$ :

- replacing graph models with trees models tends to **make logics decidable**  
recall, e.g., Rabin's theorem
- there are **powerful tools** for logics on trees (e.g. automata, games)

**Theorem.** Satisfiability in  $\mathcal{ALC}$  is  $\text{EXPTIME}$ -complete.

A simple proof is based on **type elimination** [Pratt1979]



## Data Completion—Example 3

TBox

$$A \sqsubseteq \exists r.A$$

ABox

$$\begin{array}{c} A\ a \\ \downarrow r \\ A\ b \end{array}$$

Completed ABox:

$$\begin{array}{ccccc} A\ a & \xrightarrow{r} & A\ a' & \xrightarrow{r} & A\ a'' \cdots \\ \downarrow r & & & & \\ A\ b & \xrightarrow{r} & A\ b' & \xrightarrow{r} & A\ b'' \cdots \end{array}$$

Database cannot be infinite.

$\Rightarrow$  build compact canonical model!



## ABoxes

TBox allows more complete query answers

ABox

Patient( $p$ )	finding( $p, d$ )	Inflammation( $d$ )
	location( $d, h$ )	Heart( $h$ )

TBox

Inflammation $\sqsubseteq$ Disease
HeartDisease $\doteq$ Disease $\sqcap \exists \text{location.Heart}$

Then  $p$  is an answer to query

$\exists y. \text{Patient}(\textcolor{red}{x}) \wedge \text{finding}(\textcolor{red}{x}, y) \wedge \text{HeartDisease}(y)$



The case without  $\perp$ :

- Satisfiability is **trivial**  
(Every concept satisfiable w.r.t. every TBox)
- Subsumption in  $\mathcal{EL} + \sqcup$  is still ExpTime-complete

Reduction from (un)satisfiability in  $\mathcal{EL} + \sqcup$  with  $\perp$ :

$A$  satisfiable w.r.t.  $\mathcal{T}$  iff  $\mathcal{T}' \models A \sqsubseteq A_\perp$

where  $\mathcal{T}'$  is obtained by

- replacing  $\perp$  in  $\mathcal{T}$  with  $A_\perp$  and
- adding  $\exists r.A_\perp \sqsubseteq \perp$  for all role names  $r$  used in  $\mathcal{T}$



## A Glimpse at $\mathcal{FL}_0$

To be done: describe  $C$  and  $D$  with polynomial-size TBox:

$D$  is easy:

$$\begin{aligned} L_i &\equiv \forall t.L_{i+1} \sqcap \forall f.L_{i+1} && \text{for } 1 \leq i \leq n \\ L_{n+1} &\equiv A \\ D &\equiv L_0 \end{aligned}$$

$C$  too (basically):

- for each clause  $\zeta$  of  $\varphi$ , do the construction for  $D$ , but  
drop further  $\forall t.L_{i+1} / \forall f.L_{i+1}$  when all three literals were made false
- we get concept  $\forall L.C$  with  $L$  set of truth assignments that make  $\zeta$  false
- take conjunction of all these concepts





## Query Answering

ABox

Patient( $p$ )

TBox

Patient  $\sqsubseteq$  Human

Human  $\sqsubseteq$  Male  $\sqcup$  Female

Some models:

Patient  
 $p \bullet$  Human  
Male

Patient  
 $p \bullet$  Human  
Female

Patient  
 $p \bullet$  Human  
Female  
finding  
↓  
• Meningitis

Then  $p$  is an answer to query

Human( $x$ )

Not to

Male( $x$ )

$\exists y.$ finding( $x, y$ )

