An Introduction to Description Logics

Part 2: Formal Ontologies in $\mathcal{ALC}$

Ivan Varzinczak

CRIL, Univ. Artois & CNRS
Lens, France

http://www.ijv.ovh

ESSLLI 2018

Recap: Main ingredients in formal ontologies

A common vocabulary and a shared understanding

Classes or concepts

- Describe concrete or abstract entities within the domain of interest
- E.g.: Employed student, Parent

Relations or roles

- Describe relationships between concepts or attributes of a concept
- E.g.: work for someone, being employed by someone

Instances of classes and relations

- Name objects of the domain and denote representatives of a concept
- E.g.: John, John is an employed student, John works for IBM

Recap: Why Description Logics?

Expressivity

- Concepts ✓
- Relations ✓
- Instances ✓

DLs have all one needs to formalise ontologies!

Computational properties

- Amenability to implementation ✓
- Decidability ✓
- Good trade-off between expressivity and complexity ✓

Most DL-based systems satisfy all of these!

Available tools

protégé

FaCT++

Pellet

HermiT

CEL

...

Recap: Concept language

Atomic concept names

- $\mathsf{C} =_{\text{def}} \{A_1, \ldots, A_n\}$ (Special concepts: $\top$, $\bot$)
- Intuition: basic classes of a domain of interest
- Student, Employee, Parent

Atomic role names

- $\mathsf{R} =_{\text{def}} \{r_1, \ldots, r_m\}$
- Intuition: basic relations between concepts
- worksFor, empBy

Individual names

- $\mathsf{I} =_{\text{def}} \{a_1, \ldots, a_l\}$
- Intuition: names of objects in the domain
- john, mary, ibm

Recap: Concept language

Boolean constructors

- Concept negation: $\neg$ (class complement)
- Concept conjunction: $\sqcap$ (class intersection)
- Concept disjunction: $\sqcup$ (class union)

Role restrictions

- Existential restriction: $\exists$ (at least one relationship)
- Value restriction: $\forall$ (all relationships)

Further constructors: cardinality constraints, inverse roles, ... (if needed)

Recap: Concept language

$\mathcal{ALC}$ complex concepts: $\mathcal{L}_{\mathcal{ALC}}$		Semantics: $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$
$\top,\ \bot$	(constants)	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}},\ \bot^{\mathcal{I}} = \emptyset$
A	(atomic concept)	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\neg C$	(complement)	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap D$	(conjunction)	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	(disjunction)	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists r.C$	(existential restriction)	$\{x \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(x) \cap C^{\mathcal{I}} \neq \emptyset\}$
$\forall r.C$	(value restriction)	$\{x \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(x) \subseteq C^{\mathcal{I}}\}$

Plus

Individual names a

And

$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Outline

Outline

Making Statements

DL Knowledge Bases

Entailment in DLs

Motivation

Concept language of $\mathcal{ALC}$

$\top,\ \bot$	(constants)
A	(atomic concept)
$\neg C$	(complement of C)
$C \sqcap D$	(intersection of C and D)
$C \sqcup D$	(union of C and D)
$\exists r.C$	(existential restriction)
$\forall r.C$	(value restriction)

Motivation

Concept language of $\mathcal{ALC}$

$\top,\ \bot$	(constants)
A	(atomic concept)
$\neg C$	(complement of C)
$C \sqcap D$	(intersection of C and D)
$C \sqcup D$	(union of C and D)
$\exists r.C$	(existential restriction)
$\forall r.C$	(value restriction)

Something is missing

Motivation

Concept language of $\mathcal{ALC}$

$\top, \bot$	(constants)
A	(atomic concept)
$\neg C$	(complement of C)
$C \sqcap D$	(intersection of C and D)
$C \sqcup D$	(union of C and D)
$\exists r.C$	(existential restriction)
$\forall r.C$	(value restriction)

Something is missing

- The central notion in logic: C '$\rightarrow$' D

Motivation

Concept language of $\mathcal{ALC}$

$\top,\ \bot$	(constants)
A	(atomic concept)
$\neg C$	(complement of C)
$C \sqcap D$	(intersection of C and D)
$C \sqcup D$	(union of C and D)
$\exists r.C$	(existential restriction)
$\forall r.C$	(value restriction)

Something is missing

- The central notion in logic: C '$\rightarrow$' D
- What would C '$\rightarrow$' D mean here?

Motivation

Concept language of $\mathcal{ALC}$

$\top,\ \bot$	(constants)
A	(atomic concept)
$\neg C$	(complement of C)
$C \sqcap D$	(intersection of C and D)
$C \sqcup D$	(union of C and D)
$\exists r.C$	(existential restriction)
$\forall r.C$	(value restriction)

Something is missing

- The central notion in logic: C '$\rightarrow$' D
- What would C '$\rightarrow$' D mean here? (We already have $\neg C \sqcup D$)

Motivation

Concept language of $\mathcal{ALC}$

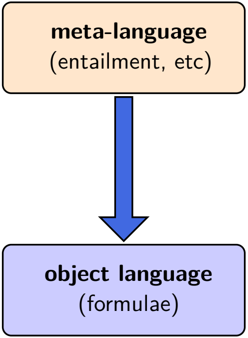
$\top,\ \bot$	(constants)
A	(atomic concept)
$\neg C$	(complement of C)
$C \sqcap D$	(intersection of C and D)
$C \sqcup D$	(union of C and D)
$\exists r.C$	(existential restriction)
$\forall r.C$	(value restriction)

Something is missing

- The central notion in logic: C '$\rightarrow$' D
- What would C '$\rightarrow$' D mean here? (We already have $\neg C \sqcup D$)
- DLs have a version of '$\rightarrow$' that is very special

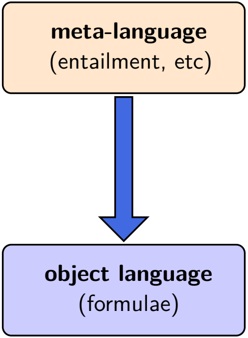
Statements

In many logics

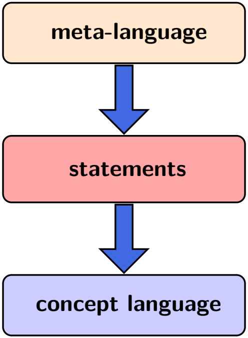


Statements

In many logics



In DLs



- Two levels of language
- Two notions of ‘entailment’
- Two notions of ‘satisfaction’

Making statements

Subsumption

- Concept inclusion
- Employed students are students
- Employed students are employees

Making statements

Subsumption

- Concept inclusion
- Employed students are students
- Employed students are employees

Instantiation or assertions

- Concept and role membership
- John is an employed student (John instantiates employed student)
- John works for IBM (John and IBM instantiate works for)

Making statements

Subsumption

- Concept inclusion
- Employed students are students
- Employed students are employees

Instantiation or assertions

- Concept and role membership
- John is an employed student (John instantiates employed student)
- John works for IBM (John and IBM instantiate works for)

Statements talk about concepts, roles and individuals

Making statements

Subsumption

- Concept inclusion
- Employed students are students
- Employed students are employees

Instantiation or assertions

- Concept and role membership
- John is an employed student (John instantiates employed student)
- John works for IBM (John and IBM instantiate works for)

Statements talk about concepts, roles and individuals

They are not concepts! They are in the 'in-between' language

Subsumption statements

$$C \sqsubseteq D$$

Intuition

- D subsumes C (or C is subsumed by D)
- C is more specific than D (or D is more general than C)
- Formalise one aspect of is-a relations

Subsumption statements

$$C \sqsubseteq D$$

Intuition

- D subsumes C (or C is subsumed by D)
- C is more specific than D (or D is more general than C)
- Formalise one aspect of is-a relations

Example

- $\mathsf{EmpStud} \sqsubseteq \mathsf{Student} \sqcap \mathsf{Employee}$, $\mathsf{Employee} \sqsubseteq \exists \mathsf{worksFor}.\top$
- $\mathsf{EmpStud} \sqsubseteq \exists \mathsf{pays}.\mathsf{Tax}$, $\mathsf{Student} \sqcap \neg \mathsf{Employee} \sqsubseteq \neg \exists \mathsf{pays}.\mathsf{Tax}$

Subsumption statements

$$C \sqsubseteq D$$

Intuition

- D subsumes C (or C is subsumed by D)
- C is more specific than D (or D is more general than C)
- Formalise one aspect of is-a relations

Example

- EmpStud $\sqsubseteq$ Student $\sqcap$ Employee, Employee $\sqsubseteq \exists$worksFor.$\top$
- EmpStud $\sqsubseteq \exists$pays.Tax, Student $\sqcap \neg$Employee $\sqsubseteq \neg\exists$pays.Tax

Central notion in DL terminologies (taxonomies)

Subsumption statements

$$C \sqsubseteq D$$

Semantics

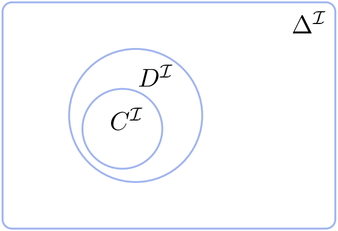
- $\mathcal{I} \Vdash C \sqsubseteq D$ ($\mathcal{I}$ satisfies $C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- First level of 'entailment': all C-objects are D-objects

Subsumption statements

$$C \sqsubseteq D$$

Semantics

- $\mathcal{I} \Vdash C \sqsubseteq D$ ($\mathcal{I}$ satisfies $C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- First level of 'entailment': all C-objects are D-objects



Subsumption statements

$$C \equiv D$$

Concept equivalence

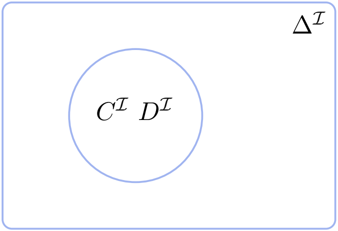
- Just an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$
- $\mathcal{I} \Vdash C \equiv D$ if $\mathcal{I} \Vdash C \sqsubseteq D$ and $\mathcal{I} \Vdash D \sqsubseteq C$
- $\mathcal{I} \Vdash C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$

Subsumption statements

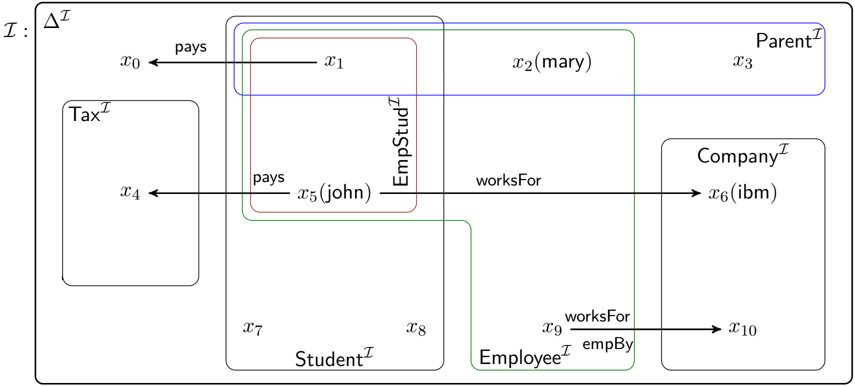
$$C \equiv D$$

Concept equivalence

- Just an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$
- $\mathcal{I} \Vdash C \equiv D$ if $\mathcal{I} \Vdash C \sqsubseteq D$ and $\mathcal{I} \Vdash D \sqsubseteq C$
- $\mathcal{I} \Vdash C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$

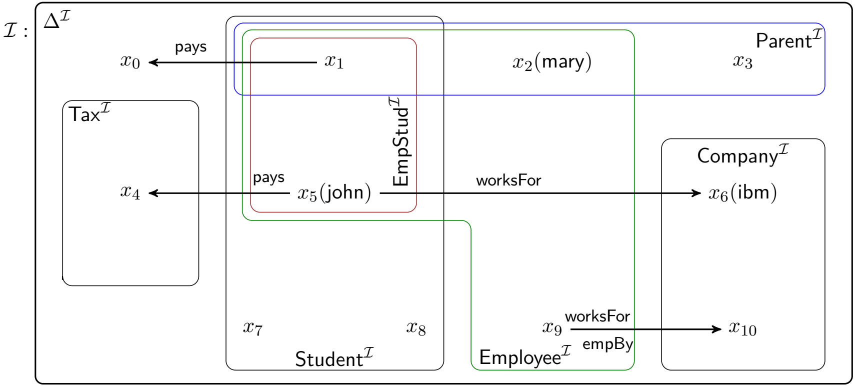


Exercise



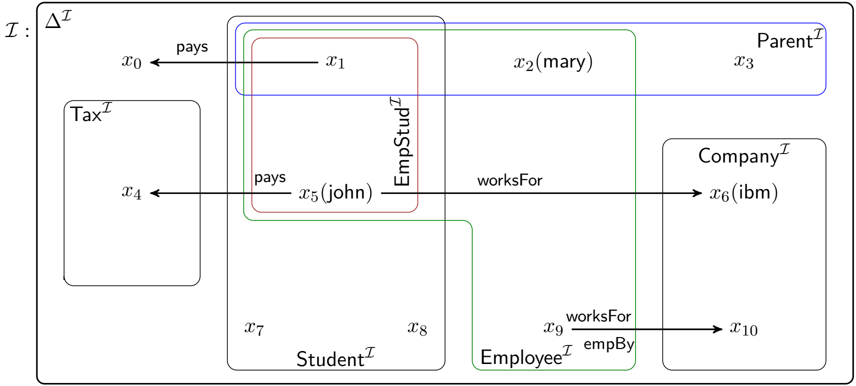
$\mathcal{I}$:
$\Delta^{\mathcal{I}}$
pays
x_0
x_1
x_2(mary)
x_3
Parent$^{\mathcal{I}}$
Tax$^{\mathcal{I}}$
EmpStud$^{\mathcal{I}}$
Company$^{\mathcal{I}}$
pays
x_4
x_5(john)
worksFor
x_6(ibm)
x_7
x_8
x_9
worksFor
empBy
x_{10}
Student$^{\mathcal{I}}$
Employee$^{\mathcal{I}}$

Exercise



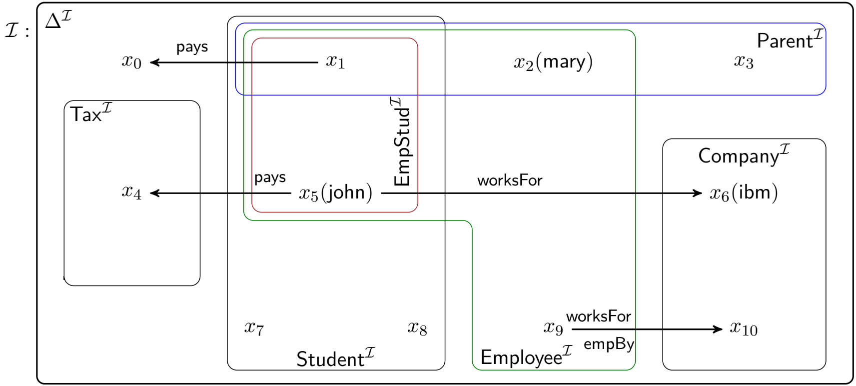
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee ?
- $\mathcal{I} \Vdash$ $\exists$worksFor.$\top$ $\sqsubseteq$ Employee ?
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq$ $\exists$worksFor.$\top$?
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ $\forall$pays.Tax ?
- $\mathcal{I} \Vdash$ Parent $\sqcap$ $\neg$Employee $\sqsubseteq$ Tax $\sqcup$ $\neg$Student ?
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq$ $\exists$pays.Tax ?
- $\mathcal{I} \Vdash$ $\exists$empBy.$\top$ $\sqsubseteq$ $\exists$worksFor.Company ?
- $\mathcal{I} \Vdash$ $\forall$pays.Tax $\sqsubseteq$ $\exists$empBy.Company ?

Exercise



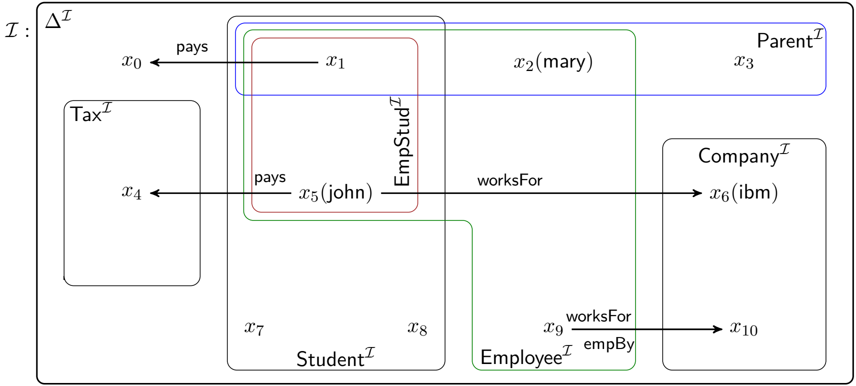
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash \exists$worksFor.$\top \sqsubseteq$ Employee ?
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq \exists$worksFor.$\top$?
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq \forall$pays.Tax ?
- $\mathcal{I} \Vdash$ Parent $\sqcap \neg$Employee $\sqsubseteq$ Tax $\sqcup \neg$Student ?
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq \exists$pays.Tax ?
- $\mathcal{I} \Vdash \exists$empBy.$\top \sqsubseteq \exists$worksFor.Company ?
- $\mathcal{I} \Vdash \forall$pays.Tax $\sqsubseteq \exists$empBy.Company ?

Exercise



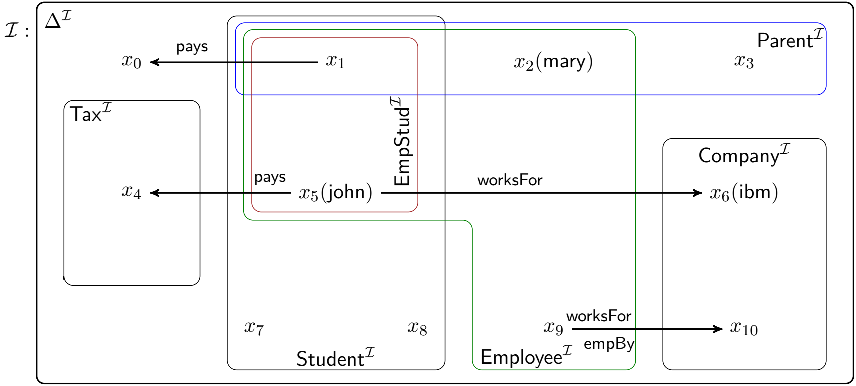
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash$ $\exists$worksFor.$\top$ $\sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq$ $\exists$worksFor.$\top$?
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ $\forall$pays.Tax ?
- $\mathcal{I} \Vdash$ Parent $\sqcap$ $\neg$Employee $\sqsubseteq$ Tax $\sqcup$ $\neg$Student ?
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq$ $\exists$pays.Tax ?
- $\mathcal{I} \Vdash$ $\exists$empBy.$\top$ $\sqsubseteq$ $\exists$worksFor.Company ?
- $\mathcal{I} \Vdash$ $\forall$pays.Tax $\sqsubseteq$ $\exists$empBy.Company ?

Exercise



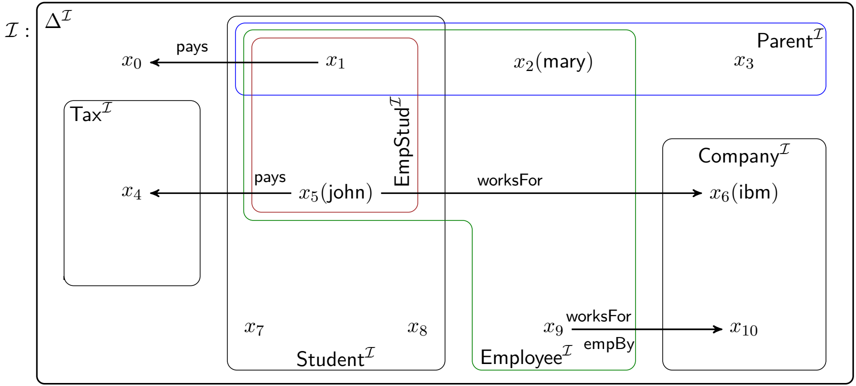
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash$ $\exists$worksFor.$\top$ $\sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq$ $\exists$worksFor.$\top$ No!
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ $\forall$pays.Tax ?
- $\mathcal{I} \Vdash$ Parent $\sqcap$ $\neg$Employee $\sqsubseteq$ Tax $\sqcup$ $\neg$Student ?
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq$ $\exists$pays.Tax ?
- $\mathcal{I} \Vdash$ $\exists$empBy.$\top$ $\sqsubseteq$ $\exists$worksFor.Company ?
- $\mathcal{I} \Vdash$ $\forall$pays.Tax $\sqsubseteq$ $\exists$empBy.Company ?

Exercise



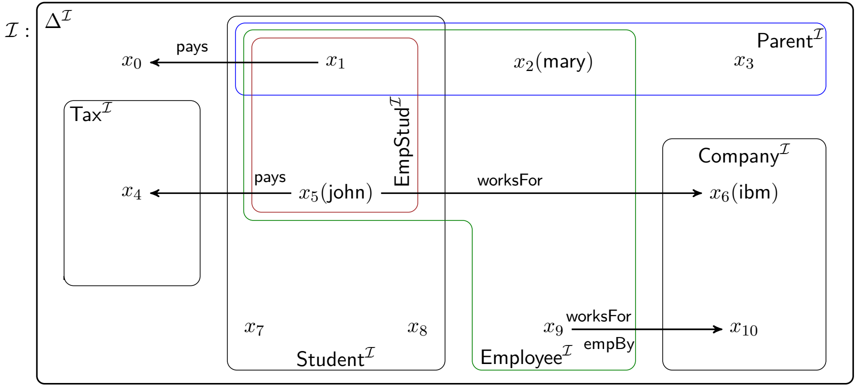
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash \exists$worksFor.$\top \sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq \exists$worksFor.$\top$ No!
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq \forall$pays.Tax No!
- $\mathcal{I} \Vdash$ Parent $\sqcap \neg$Employee $\sqsubseteq$ Tax $\sqcup \neg$Student ?
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq \exists$pays.Tax ?
- $\mathcal{I} \Vdash \exists$empBy.$\top \sqsubseteq \exists$worksFor.Company ?
- $\mathcal{I} \Vdash \forall$pays.Tax $\sqsubseteq \exists$empBy.Company ?

Exercise



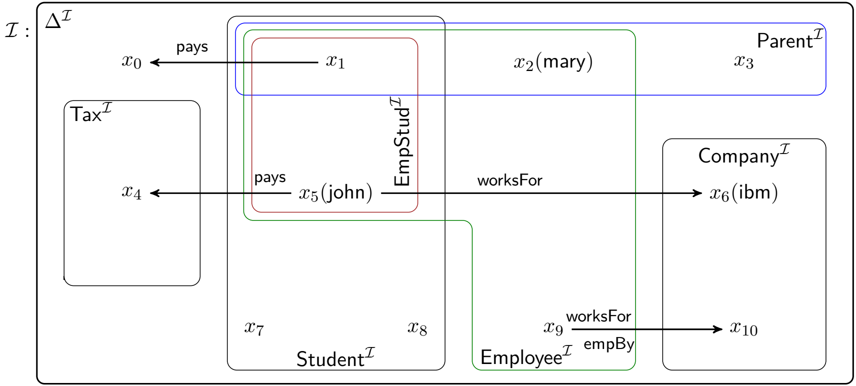
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash \exists$worksFor.$\top \sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq \exists$worksFor.$\top$ No!
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq \forall$pays.Tax No!
- $\mathcal{I} \Vdash$ Parent $\sqcap \neg$Employee $\sqsubseteq$ Tax $\sqcup \neg$Student Yep!
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq \exists$pays.Tax ?
- $\mathcal{I} \Vdash \exists$empBy.$\top \sqsubseteq \exists$worksFor.Company ?
- $\mathcal{I} \Vdash \forall$pays.Tax $\sqsubseteq \exists$empBy.Company ?

Exercise



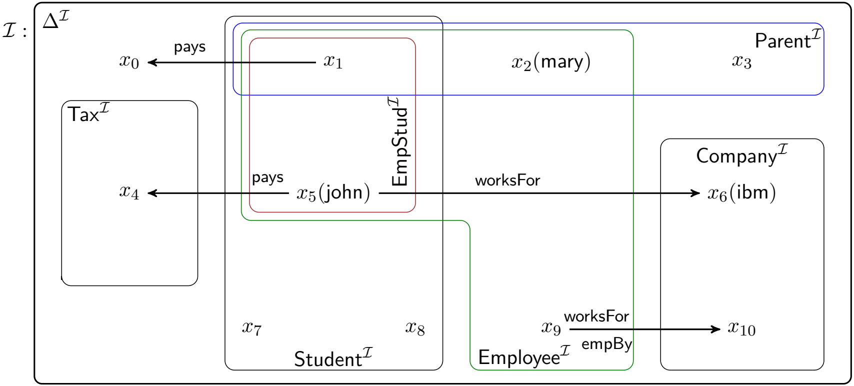
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash \exists$worksFor.$\top \sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq \exists$worksFor.$\top$ No!
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq \forall$pays.Tax No!
- $\mathcal{I} \Vdash$ Parent $\sqcap \neg$Employee $\sqsubseteq$ Tax $\sqcup \neg$Student Yep!
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq \exists$pays.Tax No!
- $\mathcal{I} \Vdash \exists$empBy.$\top \sqsubseteq \exists$worksFor.Company ?
- $\mathcal{I} \Vdash \forall$pays.Tax $\sqsubseteq \exists$empBy.Company ?

Exercise



- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash$ $\exists$worksFor.$\top$ $\sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq$ $\exists$worksFor.$\top$ No!
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ $\forall$pays.Tax No!
- $\mathcal{I} \Vdash$ Parent $\sqcap$ $\neg$Employee $\sqsubseteq$ Tax $\sqcup$ $\neg$Student Yep!
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq$ $\exists$pays.Tax No!
- $\mathcal{I} \Vdash$ $\exists$empBy.$\top$ $\sqsubseteq$ $\exists$worksFor.Company Yep!
- $\mathcal{I} \Vdash$ $\forall$pays.Tax $\sqsubseteq$ $\exists$empBy.Company ?

Exercise



- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq$ Student $\sqcap$ Employee Yep!
- $\mathcal{I} \Vdash \exists$worksFor.$\top \sqsubseteq$ Employee Yep!
- $\mathcal{I} \Vdash$ Employee $\sqsubseteq \exists$worksFor.$\top$ No!
- $\mathcal{I} \Vdash$ EmpStud $\sqsubseteq \forall$pays.Tax No!
- $\mathcal{I} \Vdash$ Parent $\sqcap \neg$Employee $\sqsubseteq$ Tax $\sqcup \neg$Student Yep!
- $\mathcal{I} \Vdash$ EmpStud $\sqcap$ Parent $\sqsubseteq \exists$pays.Tax No!
- $\mathcal{I} \Vdash \exists$empBy.$\top \sqsubseteq \exists$worksFor.Company Yep!
- $\mathcal{I} \Vdash \forall$pays.Tax $\sqsubseteq \exists$empBy.Company No!

Assertions

$$a : C \qquad\qquad (a, b) : r$$

Intuition

- a is an instance of C
- a and b are related via r (or (a, b) is an instance of r)
- Formalise another aspect of is-a relations

Assertions

$$a : C \qquad\qquad (a, b) : r$$

Intuition

- a is an instance of C
- a and b are related via r (or (a, b) is an instance of r)
- Formalise another aspect of is-a relations

Example

- $\mathsf{john} : \mathsf{EmpStud}$, $\mathsf{mary} : \mathsf{Parent} \sqcap \neg\exists \mathsf{worksFor}.\top \sqcap \neg\exists \mathsf{pays}.\mathsf{Tax}$
- $(\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor}$

Assertions

$$a : C \qquad (a, b) : r$$

Intuition

- a is an instance of C
- a and b are related via r (or (a, b) is an instance of r)
- Formalise another aspect of is-a relations

Example

- john : EmpStud, mary : Parent $\sqcap \neg\exists$worksFor.$\top \sqcap \neg\exists$pays.Tax
- (john, ibm) : worksFor (in some DLs: (john, ibm) : $\neg$empBy)

Assertions

$$a : C \qquad (a, b) : r$$

Intuition

- a is an instance of C
- a and b are related via r (or (a, b) is an instance of r)
- Formalise another aspect of is-a relations

Example

- john : EmpStud, mary : Parent $\sqcap \neg\exists$worksFor.$\top \sqcap \neg\exists$pays.Tax
- (john, ibm) : worksFor (in some DLs: (john, ibm) : $\neg$empBy)

Central notion in DL 'databases'

Assertions

$$a : C \qquad (a, b) : r$$

Semantics

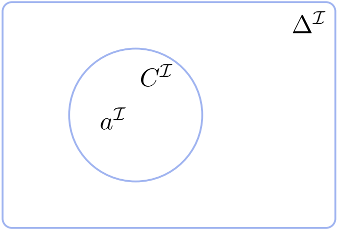
- $\mathcal{I} \Vdash a : C$ ($\mathcal{I}$ satisfies $a : C$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \Vdash (a, b) : r$ ($\mathcal{I}$ satisfies $(a, b) : r$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$
- First level of 'satisfaction': a is a 'model' of C, (a, b) is a 'model' of r

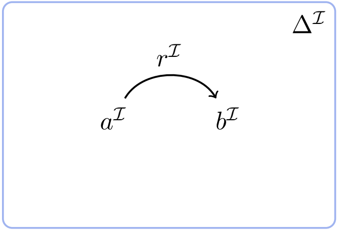
Assertions

$$a : C \qquad\qquad (a, b) : r$$

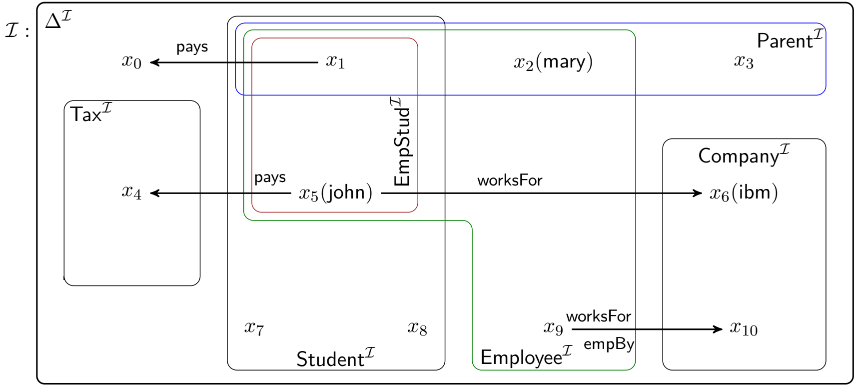
Semantics

- $\mathcal{I} \Vdash a : C$ ($\mathcal{I}$ satisfies $a : C$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \Vdash (a, b) : r$ ($\mathcal{I}$ satisfies $(a, b) : r$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$
- First level of 'satisfaction': a is a 'model' of C, (a, b) is a 'model' of r

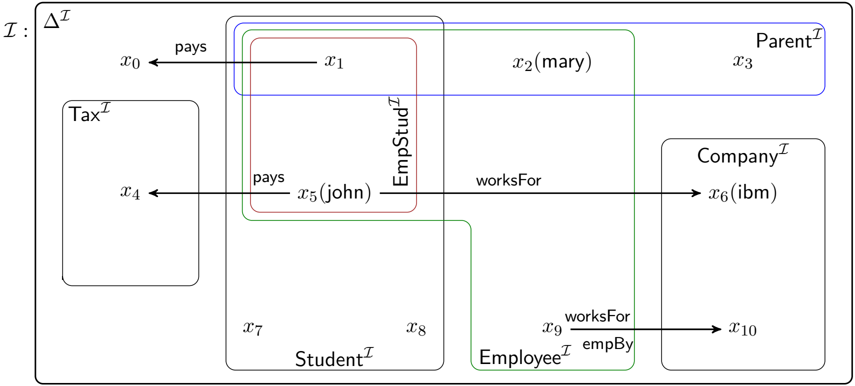




Exercise

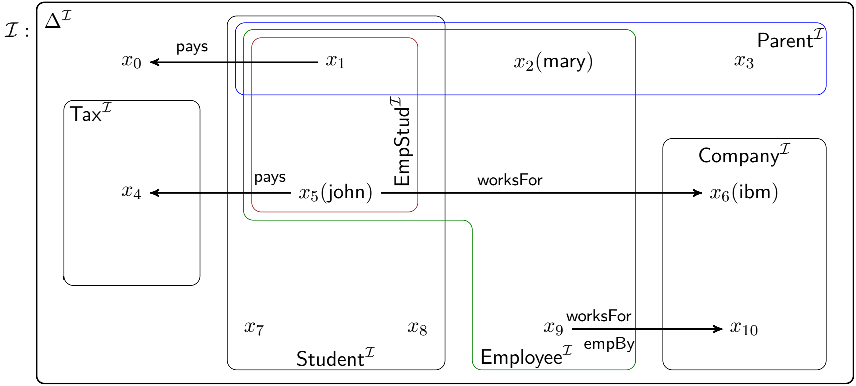

$\mathcal{I}$:
$\Delta^{\mathcal{I}}$
pays
x_0
x_1
x_2(mary)
x_3
Parent$^{\mathcal{I}}$
Tax$^{\mathcal{I}}$
EmpStud$^{\mathcal{I}}$
Company$^{\mathcal{I}}$
pays
x_4
x_5(john)
worksFor
x_6(ibm)
x_7
x_8
x_9
worksFor
empBy
x_{10}
Student$^{\mathcal{I}}$
Employee$^{\mathcal{I}}$

Exercise



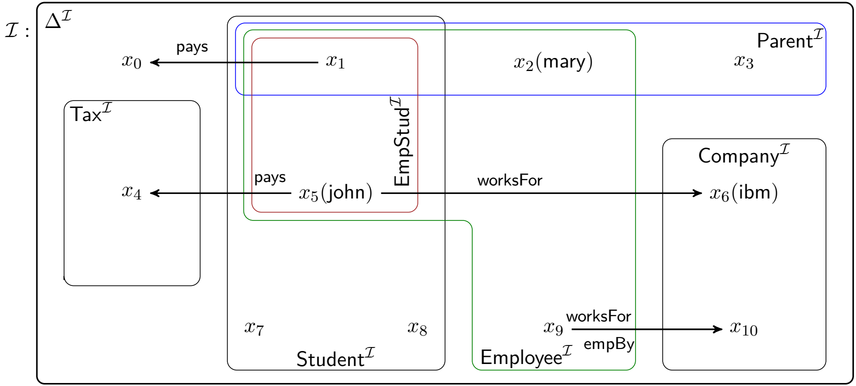
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax ?
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax ?
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy ?
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor ?
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax ?
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$?
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company ?
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$?

Exercise



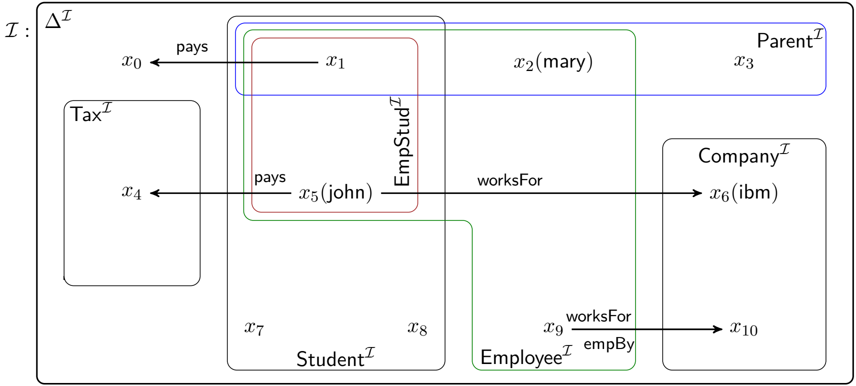
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax ?
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy ?
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor ?
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax ?
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$?
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company ?
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$?

Exercise



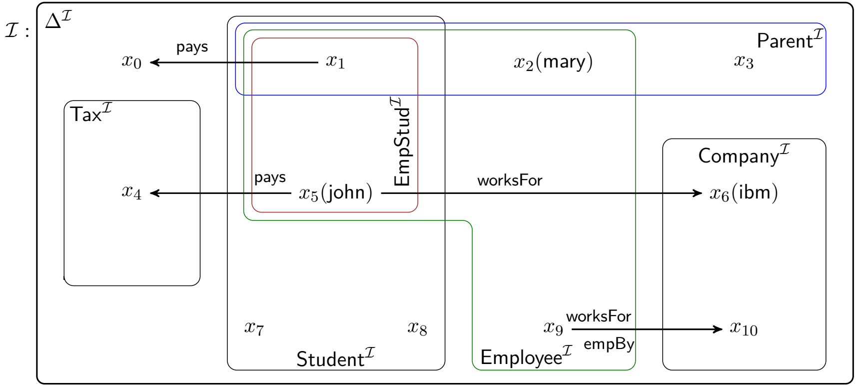
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax **Yep!**
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax **Yep!**
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy **?**
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor **?**
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax **?**
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$ **?**
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company **?**
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$ **?**

Exercise



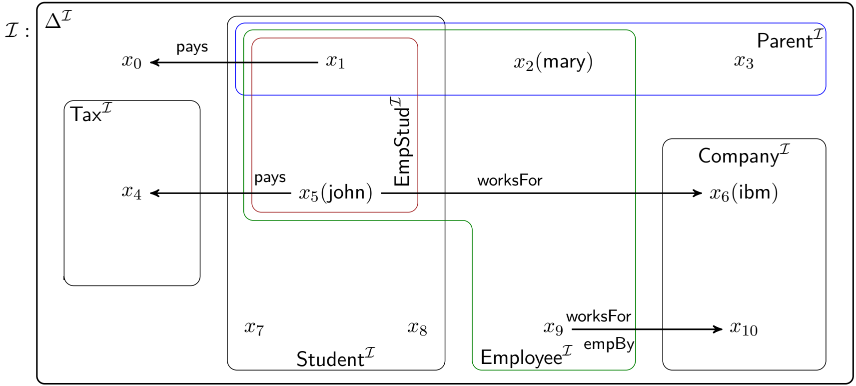
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax **Yep!**
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax **Yep!**
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy **No!**
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor **?**
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap \neg\exists$worksFor.$\top \sqcap \neg\exists$pays.Tax **?**
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$ **?**
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company **?**
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$ **?**

Exercise



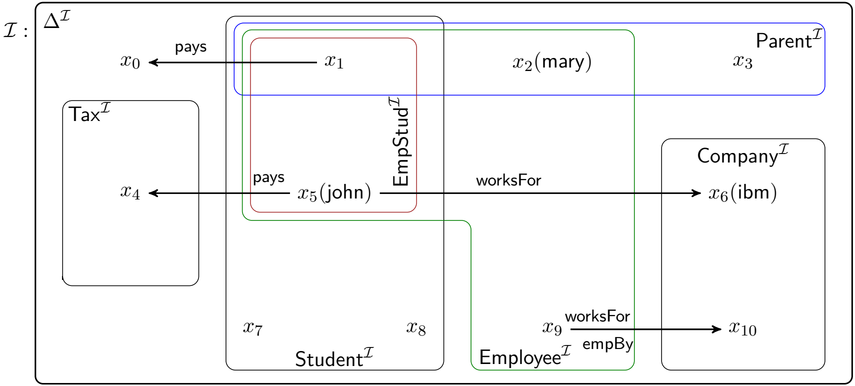
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax **Yep!**
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax **Yep!**
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy **No!**
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor **No!**
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax **?**
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$ **?**
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company **?**
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$ **?**

Exercise



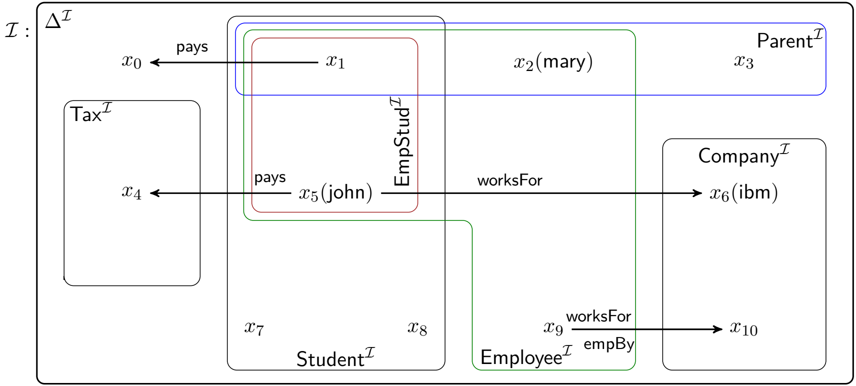
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax Yep!
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy No!
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor No!
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$?
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company ?
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$?

Exercise



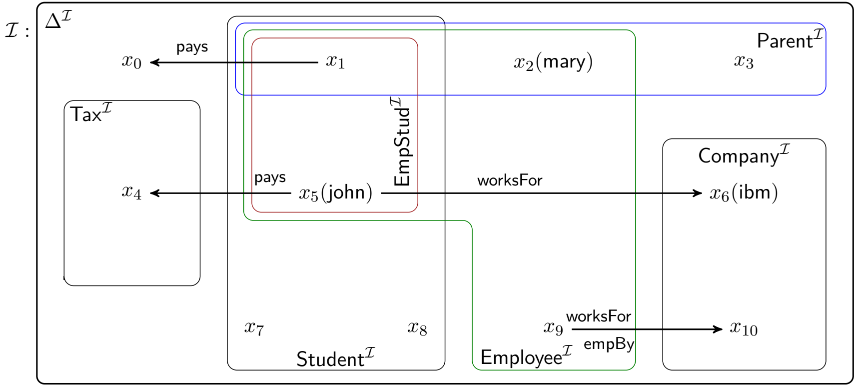
- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax Yep!
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy No!
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor No!
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$ No!
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company ?
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$?

Exercise



- $\mathcal{I} \Vdash$ john : Employee $\sqcap$ $\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax Yep!
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy No!
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor No!
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap$ $\neg\exists$worksFor.$\top$ $\sqcap$ $\neg\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap$ $\exists$empBy.$\top$ No!
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company Yep!
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$?

Exercise



- $\mathcal{I} \Vdash$ john : Employee $\sqcap\ \exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : $\forall$pays.Tax Yep!
- $\mathcal{I} \Vdash$ (mary, ibm) : empBy No!
- $\mathcal{I} \Vdash$ (ibm, john) : worksFor No!
- $\mathcal{I} \Vdash$ mary : Parent $\sqcap \neg\exists$worksFor.$\top \sqcap \neg\exists$pays.Tax Yep!
- $\mathcal{I} \Vdash$ mary : Employee $\sqcap\ \exists$empBy.$\top$ No!
- $\mathcal{I} \Vdash$ john : $\forall$empBy.Company Yep!
- $\mathcal{I} \Vdash$ john : $\exists$worksFor.$\forall$pays.$\bot$ Yep!

Subsumptions and assertions

Validity

- Let α denote a statement
- $\models \alpha$ (α is valid) if $\mathcal{I} \Vdash \alpha$ for every $\mathcal{I}$

Subsumptions and assertions

Validity

- Let α denote a statement
- $\models \alpha$ (α is valid) if $\mathcal{I} \Vdash \alpha$ for every $\mathcal{I}$

Example

- $\models \neg(C \sqcap D) \equiv (\neg C \sqcup \neg D)$
- $\models \forall r.(C \sqcap D) \sqsubseteq \forall r.C$
- $\not\models \exists r.\top \sqsubseteq \exists r.C$
- $\models a : C \sqcup \neg C$
- $\models \neg(C \sqcup D) \equiv (\neg C \sqcap \neg D)$
- $\not\models \forall r.C \sqsubseteq \forall r.(C \sqcap D)$
- $\models \exists r.C \sqsubseteq \exists r.\top$
- $\not\models (a, b) : r$

Subsumptions and assertions

Validity

- Let α denote a statement
- $\models \alpha$ (α is valid) if $\mathcal{I} \Vdash \alpha$ for every $\mathcal{I}$

Example

- $\models \neg(C \sqcap D) \equiv (\neg C \sqcup \neg D)$
- $\models \neg(C \sqcup D) \equiv (\neg C \sqcap \neg D)$
- $\models \forall r.(C \sqcap D) \sqsubseteq \forall r.C$
- $\not\models \forall r.C \sqsubseteq \forall r.(C \sqcap D)$
- $\not\models \exists r.\top \sqsubseteq \exists r.C$
- $\models \exists r.C \sqsubseteq \exists r.\top$
- $\models a : C \sqcup \neg C$
- $\not\models (a, b) : r$

Watch out: Statements can be valid; concepts cannot!

Outline

Making Statements

DL Knowledge Bases

Entailment in DLs

TBoxes and ABoxes

Intensional knowledge

- Set of subsumption statements
- Intuition: provide definitions of concepts (a terminology)
- Called the TBox (terminological box). Notation: $\mathcal{T}$

TBoxes and ABoxes

Intensional knowledge

- Set of subsumption statements
- Intuition: provide definitions of concepts (a terminology)
- Called the TBox (terminological box). Notation: $\mathcal{T}$

Extensional knowledge

- Concept and role assertions
- Intuition: provide an instantiation of concepts and roles (a 'database')
- Called the ABox (assertion box). Notation: $\mathcal{A}$

TBoxes and ABoxes

Intensional knowledge

- Set of subsumption statements
- Intuition: provide definitions of concepts (a terminology)
- Called the TBox (terminological box). Notation: $\mathcal{T}$

Extensional knowledge

- Concept and role assertions
- Intuition: provide an instantiation of concepts and roles (a 'database')
- Called the ABox (assertion box). Notation: $\mathcal{A}$

Definition (Knowledge base)

A DL knowledge base (a.k.a. ontology) is a tuple $\mathcal{KB} =_{\text{def}} \langle \mathcal{T}, \mathcal{A} \rangle$

Knowledge bases

Example (The student ontology in DL)

$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{EmpStud} \equiv \mathsf{Student} \sqcap \mathsf{Employee}, \\ \mathsf{Student} \sqcap \neg \mathsf{Employee} \sqsubseteq \neg \exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \neg \mathsf{Parent} \sqsubseteq \exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \mathsf{Parent} \sqsubseteq \neg \exists \mathsf{pays}.\mathsf{Tax}, \\ \exists \mathsf{worksFor}.\mathsf{Company} \sqsubseteq \mathsf{Employee} \end{array} \right\}$$

$$\mathcal{A} = \left\{ \begin{array}{c} \mathsf{ibm} : \mathsf{Company}, \\ \mathsf{mary} : \mathsf{Parent}, \\ \mathsf{john} : \mathsf{EmpStud}, \\ (\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor} \end{array} \right\}$$

classes
relations
individuals

Knowledge bases

Semantics

- $\mathcal{I} \Vdash \mathcal{T}$ if $\mathcal{I} \Vdash C \sqsubseteq D$ for every $C \sqsubseteq D \in \mathcal{T}$
- $\mathcal{I} \Vdash \mathcal{A}$ if:
  - $\mathcal{I} \Vdash a : C$ for every $a : C \in \mathcal{A}$, and
  - $\mathcal{I} \Vdash (a, b) : r$ for every $(a, b) : r \in \mathcal{A}$

Knowledge bases

Semantics

- $\mathcal{I} \Vdash \mathcal{T}$ if $\mathcal{I} \Vdash C \sqsubseteq D$ for every $C \sqsubseteq D \in \mathcal{T}$
- $\mathcal{I} \Vdash \mathcal{A}$ if:
  - $\mathcal{I} \Vdash a : C$ for every $a : C \in \mathcal{A}$, and
  - $\mathcal{I} \Vdash (a, b) : r$ for every $(a, b) : r \in \mathcal{A}$

Moreover

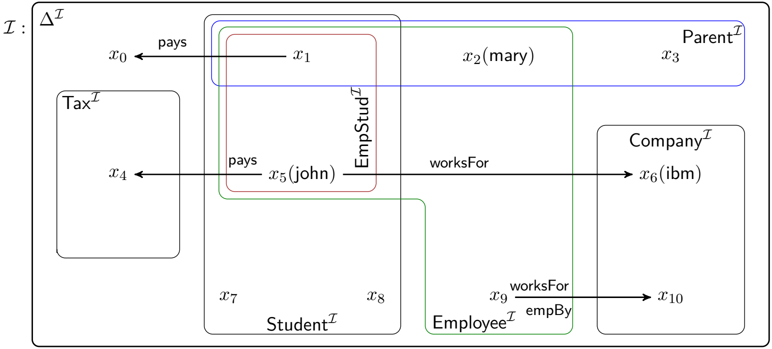
- If $\mathcal{I} \Vdash \mathcal{T}$, we say $\mathcal{I}$ is a model of $\mathcal{T}$
- If $\mathcal{I} \Vdash \mathcal{A}$, we say $\mathcal{I}$ is a model of $\mathcal{A}$
- If $\mathcal{I} \Vdash \mathcal{T} \cup \mathcal{A}$, then $\mathcal{I}$ is a model of $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$
- $\mathcal{KB}$ is satisfiable if it has a model

Exercise

- Let $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{EmpStud} \equiv \mathsf{Student} \sqcap \mathsf{Employee}, \\ \mathsf{Student} \sqcap \neg\mathsf{Employee} \sqsubseteq \neg\exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \neg\mathsf{Parent} \sqsubseteq \exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \mathsf{Parent} \sqsubseteq \neg\exists \mathsf{pays}.\mathsf{Tax}, \\ \exists \mathsf{worksFor}.\mathsf{Company} \sqsubseteq \mathsf{Employee} \end{array} \right\} \qquad \mathcal{A} = \left\{ \begin{array}{c} \mathsf{ibm} : \mathsf{Company}, \\ \mathsf{mary} : \mathsf{Parent}, \\ \mathsf{john} : \mathsf{EmpStud}, \\ (\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor} \end{array} \right\}$$

- Is the interpretation $\mathcal{I}$ below a model of $\mathcal{KB}$?



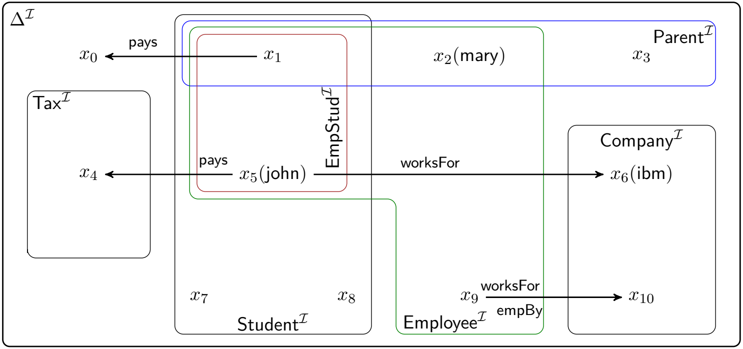
Exercise

- Let $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

$$\mathcal{T} = \left\{ \begin{array}{c} \text{EmpStud} \equiv \text{Student} \sqcap \text{Employee}, \\ \text{Student} \sqcap \neg\text{Employee} \sqsubseteq \neg\exists\text{pays}.\text{Tax}, \\ \text{EmpStud} \sqcap \neg\text{Parent} \sqsubseteq \exists\text{pays}.\text{Tax}, \\ \text{EmpStud} \sqcap \text{Parent} \sqsubseteq \neg\exists\text{pays}.\text{Tax}, \\ \exists\text{worksFor}.\text{Company} \sqsubseteq \text{Employee} \end{array} \right\} \qquad \mathcal{A} = \left\{ \begin{array}{c} \text{ibm} : \text{Company}, \\ \text{mary} : \text{Parent}, \\ \text{john} : \text{EmpStud}, \\ (\text{john}, \text{ibm}) : \text{worksFor} \end{array} \right\}$$

- Is the interpretation $\mathcal{I}$ below a model of $\mathcal{KB}$? Yep!

$\mathcal{I}$:



Exercise

- Let $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{EmpStud} \equiv \mathsf{Student} \sqcap \mathsf{Employee}, \\ \mathsf{Student} \sqcap \neg\mathsf{Employee} \sqsubseteq \neg\exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \neg\mathsf{Parent} \sqsubseteq \exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \mathsf{Parent} \sqsubseteq \neg\exists \mathsf{pays}.\mathsf{Tax}, \\ \exists \mathsf{worksFor}.\mathsf{Company} \sqsubseteq \mathsf{Employee} \end{array} \right\} \qquad \mathcal{A} = \left\{ \begin{array}{c} \mathsf{ibm} : \mathsf{Company}, \\ \mathsf{mary} : \mathsf{Parent}, \\ \mathsf{john} : \mathsf{EmpStud}, \\ (\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor} \end{array} \right\}$$

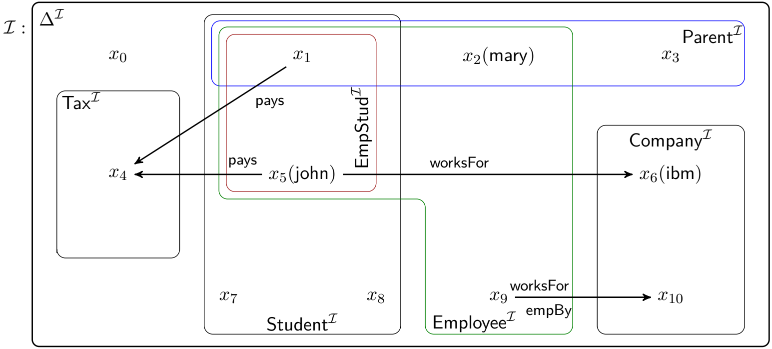
- Find a counter-model for this knowledge base

Exercise

- Let $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{EmpStud} \equiv \mathsf{Student} \sqcap \mathsf{Employee}, \\ \mathsf{Student} \sqcap \neg\mathsf{Employee} \sqsubseteq \neg\exists\mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \neg\mathsf{Parent} \sqsubseteq \exists\mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \mathsf{Parent} \sqsubseteq \neg\exists\mathsf{pays}.\mathsf{Tax}, \\ \exists\mathsf{worksFor}.\mathsf{Company} \sqsubseteq \mathsf{Employee} \end{array} \right\} \qquad \mathcal{A} = \left\{ \begin{array}{c} \mathsf{ibm} : \mathsf{Company}, \\ \mathsf{mary} : \mathsf{Parent}, \\ \mathsf{john} : \mathsf{EmpStud}, \\ (\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor} \end{array} \right\}$$

- Find a counter-model for this knowledge base



Translating KBs. . .

. . . in FOL

Extend τ to a mapping from $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$ to conjunctions of FOL formulae

$$\tau(C \sqsubseteq D) = \forall x.[\tau(C, x) \rightarrow \tau(D, x)]$$

$$\tau(C \equiv D) = \forall x.[\tau(C, x) \leftrightarrow \tau(D, x)]$$

$$\tau(a : C) = \tau(C, c_a)$$

$$\tau((a, b) : r) = p_r(c_a, c_b)$$

Translating KBs...

...in FOL

Extend τ to a mapping from $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$ to conjunctions of FOL formulae

$$\tau(C \sqsubseteq D) = \forall x.[\tau(C, x) \rightarrow \tau(D, x)]$$

$$\tau(C \equiv D) = \forall x.[\tau(C, x) \leftrightarrow \tau(D, x)]$$

$$\tau(a : C) = \tau(C, c_a)$$

$$\tau((a, b) : r) = p_r(c_a, c_b)$$

Then

- $\tau(\mathcal{KB}) = (\bigwedge_{\alpha \in \mathcal{T}} \tau(\alpha)) \wedge (\bigwedge_{\alpha \in \mathcal{A}} \tau(\alpha))$

Translating KBs...

...in FOL

Extend τ to a mapping from $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$ to conjunctions of FOL formulae

$$\tau(C \sqsubseteq D) = \forall x.[\tau(C, x) \rightarrow \tau(D, x)]$$
$$\tau(C \equiv D) = \forall x.[\tau(C, x) \leftrightarrow \tau(D, x)]$$
$$\tau(a : C) = \tau(C, c_a)$$
$$\tau((a, b) : r) = p_r(c_a, c_b)$$

Then

- $\tau(\mathcal{KB}) = (\bigwedge_{\alpha \in \mathcal{T}} \tau(\alpha)) \wedge (\bigwedge_{\alpha \in \mathcal{A}} \tau(\alpha)) \wedge (\bigwedge_{\text{distinct } a,b \text{ occurring in } \mathcal{A}} c_a \neq c_b)$

Translating KBs...

...in FOL

Extend τ to a mapping from $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$ to conjunctions of FOL formulae

$$\tau(C \sqsubseteq D) = \forall x.[\tau(C, x) \rightarrow \tau(D, x)]$$
$$\tau(C \equiv D) = \forall x.[\tau(C, x) \leftrightarrow \tau(D, x)]$$
$$\tau(a : C) = \tau(C, c_a)$$
$$\tau((a, b) : r) = p_r(c_a, c_b)$$

Then

- $\tau(\mathcal{KB}) = (\bigwedge_{\alpha \in \mathcal{T}} \tau(\alpha)) \wedge (\bigwedge_{\alpha \in \mathcal{A}} \tau(\alpha)) \wedge (\bigwedge_{\text{distinct } a,b \text{ occurring in } \mathcal{A}} c_a \neq c_b)$

Theorem

$\mathcal{KB}$ is satisfiable iff $\tau(\mathcal{KB})$ is FOL-satisfiable

Translating KBs. . .

. . . in modal logic

We need a universal modality and nominals

Then

We extend η to a mapping from statements to formulae of hybrid logic

$$\eta(C \sqsubseteq D) = \Box(\eta(C) \rightarrow \eta(D))$$

$$\eta(C \equiv D) = \Box(\eta(C) \leftrightarrow \eta(D))$$

$$\eta(a : C) = @_{n_a}\eta(C)$$

$$\eta((a, b) : r) = @_{n_a}\langle i_r \rangle n_b$$

Translating KBs. . .

. . . in modal logic

We need a universal modality and nominals

Then

We extend η to a mapping from statements to formulae of hybrid logic

$$\eta(C \sqsubseteq D) = \Box(\eta(C) \rightarrow \eta(D))$$
$$\eta(C \equiv D) = \Box(\eta(C) \leftrightarrow \eta(D))$$
$$\eta(a : C) = @_{n_a}\eta(C)$$
$$\eta((a, b) : r) = @_{n_a}\langle i_r\rangle n_b$$

Theorem

$\mathcal{KB}$ is satisfiable iff $\eta(\mathcal{KB})$ is modally satisfiable

Outline

Making Statements

DL Knowledge Bases

Entailment in DLs

What does follow from a KB?

Example

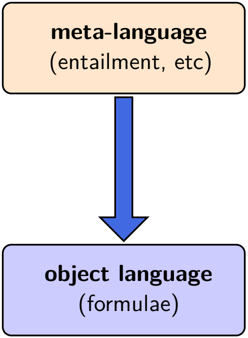
- Is an employed parent an employee who is also a student?
- Is being an employee the same as being employed by someone?
- Is Mary an employed parent?
- Is being an employee more general than being a employed student?
- Is there anybody who is a student and a parent at the same time?
- Is my knowledge base consistent?
- Tell me, briefly, what John is.
- Who are the employed students who work for companies?

What does follow from a KB?

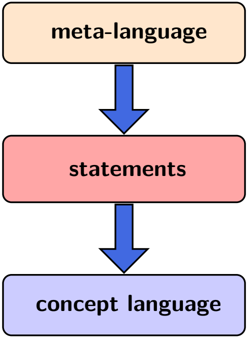
Entailment from KBs

- Defined on the level of statements (not concepts)
- Remember:

In many logics



In DLs



What does follow from a KB?

Entailment from KBs

- Given a TBox $\mathcal{T}$, what other subsumptions follow?
- Given an ABox $\mathcal{A}$, what other assertions follow?
- Given a knowledge base $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, what statements follow from it?

What does follow from a KB?

Entailment from KBs

- Given a TBox $\mathcal{T}$, what other subsumptions follow?
- Given an ABox $\mathcal{A}$, what other assertions follow?
- Given a knowledge base $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, what statements follow from it?

Obvious definition of entailment

- $\mathcal{T} \models \alpha$ if $\mathcal{I} \Vdash \alpha$ for every $\mathcal{I}$ s.t. $\mathcal{I} \Vdash \mathcal{T}$
- $\mathcal{A} \models \alpha$ if $\mathcal{I} \Vdash \alpha$ for every $\mathcal{I}$ s.t. $\mathcal{I} \Vdash \mathcal{A}$
- $\mathcal{KB} \models \alpha$ if $\mathcal{I} \Vdash \alpha$ for every $\mathcal{I}$ s.t. $\mathcal{I} \Vdash \mathcal{KB}$

What does follow from a KB?

Example

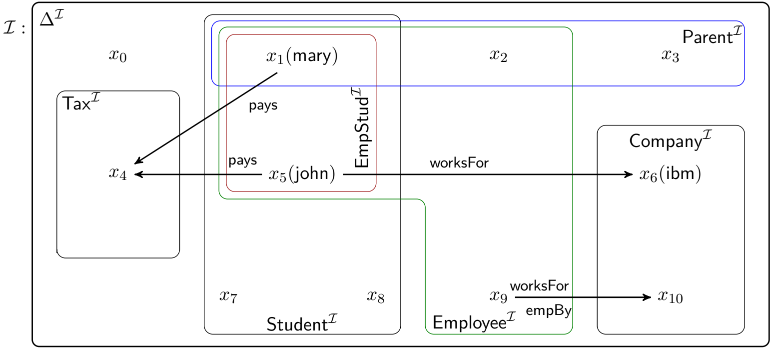
$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{EmpStud} \equiv \mathsf{Student} \sqcap \mathsf{Employee}, \\ \mathsf{Student} \sqcap \neg\mathsf{Employee} \sqsubseteq \neg\exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \neg\mathsf{Parent} \sqsubseteq \exists \mathsf{pays}.\mathsf{Tax}, \\ \mathsf{EmpStud} \sqcap \mathsf{Parent} \sqsubseteq \neg\exists \mathsf{pays}.\mathsf{Tax}, \\ \exists \mathsf{worksFor}.\mathsf{Company} \sqsubseteq \mathsf{Employee} \end{array} \right\} \quad \mathcal{A} = \left\{ \begin{array}{c} \mathsf{ibm} : \mathsf{Company}, \\ \mathsf{mary} : \mathsf{Parent}, \\ \mathsf{john} : \mathsf{EmpStud}, \\ (\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor} \end{array} \right\}$$

- $\mathcal{KB} \models \mathsf{Student} \sqcap \exists \mathsf{worksFor}.\mathsf{Company} \sqcap \neg\mathsf{Parent} \sqsubseteq \mathsf{EmpStud} \sqcap \exists \mathsf{pays}.\mathsf{Tax}$
- $\mathcal{KB} \models \mathsf{john} : \mathsf{Student} \sqcap \exists \mathsf{worksFor}.\mathsf{Company}$
- $\mathcal{KB} \not\models \mathsf{mary} : \neg\exists \mathsf{pays}.\mathsf{Tax}$
- $\mathcal{KB} \not\models \mathsf{Employee} \sqsubseteq \exists \mathsf{empBy}.\mathsf{Company}$

What does follow from a KB?

Example

- $\mathcal{KB} \not\models$ mary : $\neg\exists$pays.Tax
- $\mathcal{KB} \not\models$ Employee $\sqsubseteq \exists$empBy.Company



Open- v. closed-world assumption

Closed-world assumption (CWA)

- $\mathcal{KB}$ contains all information
- Non-derivable statements are assumed to be false

Open- v. closed-world assumption

Closed-world assumption (CWA)

- $\mathcal{KB}$ contains all information
- Non-derivable statements are assumed to be false

Open-world assumption (OWA)

- $\mathcal{KB}$ may be incomplete
- Truth of non-derivable statements is just unknown

Open- v. closed-world assumption

Closed-world assumption (CWA)

- $\mathcal{KB}$ contains all information
- Non-derivable statements are assumed to be false

Open-world assumption (OWA)

- $\mathcal{KB}$ may be incomplete
- Truth of non-derivable statements is just unknown

Example

$\{(\mathsf{john}, \mathsf{ibm}) : \mathsf{worksFor},\ \mathsf{ibm} : \mathsf{Company})\} \models \mathsf{john} : \forall \mathsf{worksFor}.\mathsf{Company}$?

- In Prolog: “Yep!”
- In DL-based systems: “Uh, I don't know . . . ”

DLs are (very) classical

Definition (Consequence operator)

Given $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, the set of all consequences of $\mathcal{KB}$ is

$$Cn(\mathcal{KB}) =_{\text{def}} \{\alpha \mid \mathcal{KB} \models \alpha\}$$

DLs are (very) classical

Definition (Consequence operator)

Given $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, the set of all consequences of $\mathcal{KB}$ is

$$Cn(\mathcal{KB}) =_{\text{def}} \{\alpha \mid \mathcal{KB} \models \alpha\}$$

Theorem

$Cn(\cdot)$ satisfies the following properties:

- $\mathcal{KB} \subseteq Cn(\mathcal{KB})$ (Inclusion)
- $Cn(\mathcal{KB}) = Cn(Cn(\mathcal{KB}))$ (Idempotency)
- If $\mathcal{KB}_1 \subseteq \mathcal{KB}_2$, then $Cn(\mathcal{KB}_1) \subseteq Cn(\mathcal{KB}_2)$ (Monotonicity)

DLs are (very) classical

Definition (Consequence operator)

Given $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, the set of all consequences of $\mathcal{KB}$ is

$$Cn(\mathcal{KB}) =_{\text{def}} \{\alpha \mid \mathcal{KB} \models \alpha\}$$

Theorem

$Cn(\cdot)$ satisfies the following properties:

- $\mathcal{KB} \subseteq Cn(\mathcal{KB})$ (Inclusion)
- $Cn(\mathcal{KB}) = Cn(Cn(\mathcal{KB}))$ (Idempotency)
- If $\mathcal{KB}_1 \subseteq \mathcal{KB}_2$, then $Cn(\mathcal{KB}_1) \subseteq Cn(\mathcal{KB}_2)$ (Monotonicity)

Hence, $Cn(\cdot)$ is a Tarskian consequence operator

Epilogue

Summary

- Intensional and extensional knowledge
- Specifying DL knowledge bases
  - TBox: categories
  - ABox: partial view of the world
- What follows from a DL ontology

Epilogue

Summary

- Intensional and extensional knowledge
- Specifying DL knowledge bases
  - TBox: categories
  - ABox: partial view of the world
- What follows from a DL ontology

What next?

- Reasoning with DL ontologies