Ontology Engineering

Lecture 3: Description Logics

Maria Keet

email: mkeet@cs.uct.ac.za

home: http://www.meteck.org

Department of Computer Science
University of Cape Town, South Africa

Semester 2, Block I, 2019

Outline

Outline

1. Introduction
2. Basic DL: $\mathcal{ALC}$
   - Syntax
   - Semantics
3. DL and FOL
4. Reasoning services
   - Standard services
   - Techniques

Why description logics

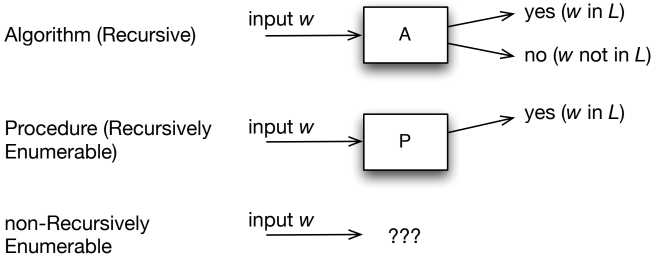
- Just saw FOL, so why the hassle of looking at another logic?

Why description logics

- Just saw FOL, so why the hassle of looking at another logic?
- Full FOL is undecidable, which is bad news for scalable implementations

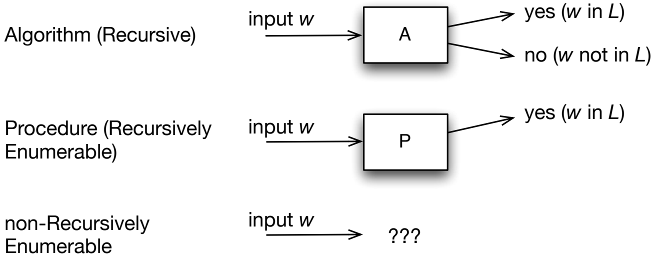
Why description logics

- Just saw FOL, so why the hassle of looking at another logic?
- Full FOL is undecidable, which is bad news for scalable implementations



Why description logics

- Just saw FOL, so why the hassle of looking at another logic?
- Full FOL is undecidable, which is bad news for scalable implementations



- Multiple applications (recall lecture 1) use OWL, which is actually DL-for-computational-use (except for OWL full)
- Need to grasp basics of the language so as to understand what's going on when developing an ontology (the reasoner output really is not magic)

What are DLs?

- A structured fragment of FOL
- Different notation, but very same ideas as we've seen in previous lecture
- (we'll get back to the 'fragment' aspect later)

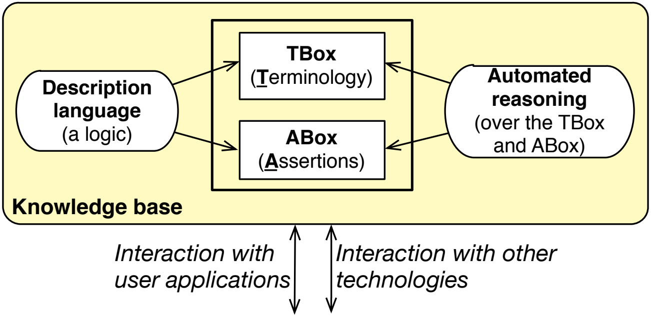
What are DLs?

- A structured fragment of FOL
- Different notation, but very same ideas as we've seen in previous lecture
- (we'll get back to the 'fragment' aspect later)
- (Any (basic) Description Logic is a subset of $\mathcal{L}_3$, i.e., the function-free FOL using only at most three variable names)
- Representation is at the predicate level: no variables are present in the notation (formalism)

What are DLs?

- A structured fragment of FOL
- Different notation, but very same ideas as we've seen in previous lecture
- (we'll get back to the 'fragment' aspect later)
- (Any (basic) Description Logic is a subset of $\mathcal{L}_3$, i.e., the function-free FOL using only at most three variable names)
- Representation is at the predicate level: no variables are present in the notation (formalism)
- Provide theories and systems for declaratively expressing structured information and for accessing and reasoning with it.
- Used for, a.o.: terminologies and ontologies, formal conceptual data modelling, information integration, ...

Description Logic knowledge base



Outline

$\mathcal{ALC}$ syntax

- Concepts denoting entity types/classes/unary predicates/universals, including top $\top$ and bottom $\bot$;
  Example: (primitive, atomic): `Book`, `Course`
- Roles denoting relationships/associations/n-ary predicates/properties;
  Example[1]: `ENROLLED`, `READS`
- Constructors: 'and' $\sqcap$, 'or' $\sqcup$, and 'not' $\neg$; quantifiers 'for all' (each) $\forall$ and 'exists' (at least one/some) $\exists$
- Individuals (objects)
  Example: $\texttt{Student}(\texttt{Mandla})$, $\texttt{Mother}(\texttt{Sally})$, $\neg\texttt{Student}(\texttt{Sally})$, $\texttt{ENROLLED}(\texttt{Mandla}, \texttt{CS101/19/2})$

[1] Capitalisation for roles for notational clarity, but not required

$\mathcal{ALC}$ syntax

- Complex concepts using constructors
  - Let C and D be concept names, R a role name, then
  - $\neg C$, $C \sqcap D$, and $C \sqcup D$ are concepts, and
  - $\forall R.C$ and $\exists R.C$ are concepts

$\mathcal{ALC}$ syntax

- Complex concepts using constructors
  - Let C and D be concept names, R a role name, then
  - $\neg C$, $C \sqcap D$, and $C \sqcup D$ are concepts, and
  - $\forall R.C$ and $\exists R.C$ are concepts
- Examples:
  - $\texttt{Student} \sqsubseteq \exists\texttt{ENROLLED}.(\texttt{Course} \sqcup \texttt{DegreeProgramme})$
    this is a *primitive concept*
  - $\texttt{Mother} \sqsubseteq \texttt{Woman} \sqcap \exists\texttt{PARENTOF}.\texttt{Person}$
  - $\texttt{Parent} \equiv (\texttt{Male} \sqcup \texttt{Female}) \sqcap \exists\texttt{PARENTOF}.\texttt{Mammal} \sqcap$
    $\exists\texttt{CARESFOR}.\texttt{Mammal}$
    this is a *defined concept*

$\mathcal{ALC}$ syntax

- Domain and range restrictions of roles
- Or: specifying what kind of object the first (domain) and the second (range) object participating in the role has to be.

$\mathcal{ALC}$ syntax

- Domain and range restrictions of roles
- Or: specifying what kind of object the first (domain) and the second (range) object participating in the role has to be.
- e.g., SONOF: the domain surely has to be male, and the range is a parent:
  - $\exists \texttt{SONOF}.\top \sqsubseteq \texttt{Male}$: "any object that has an outgoing relation SONOF is a male"
  - $\top \sqsubseteq \forall \texttt{SONOF}.\texttt{Parent}$: "all objects that have an incoming relation SONOF are a parent"
    $\exists \texttt{SONOF}^{-}.\top \sqsubseteq \texttt{Parent}$: "the domain of the inverse of SONOF (i.e., range of SONOF) is a parent"

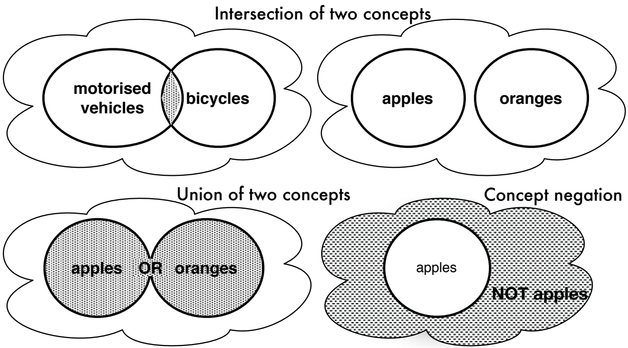
Semantics of $\mathcal{ALC}$

- Model-theoretic semantics
- Domain Δ is a non-empty set of objects
- Interpretation: $\cdot^{\mathcal{I}}$ is the *interpretation function*, domain $\Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every concept name A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every role name R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every individual name a to elements of $\Delta^{\mathcal{I}}$: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- Note: $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$

Semantics of $\mathcal{ALC}$ (2/3)

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$

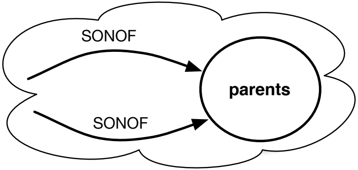
The cloud-shape is our domain of interpretation with objects



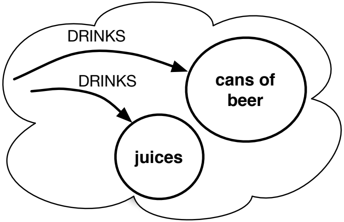
Semantics of $\mathcal{ALC}$

- C and D are concepts, R a role
- $(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y)\}$
- $(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\}$

The cloud-shape is our domain of interpretation with objects



All SONOF relations relate to a parent



At least one DRINKS relation relates to a can of beer, but there may be other DRINKS relations to other drinks, such as juices

Semantics of $\mathcal{ALC}$

- C and D are concepts, R a role, a and b are individuals
- An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$

Semantics of $\mathcal{ALC}$

- C and D are concepts, R a role, a and b are individuals
- An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$
- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Semantics of $\mathcal{ALC}$

- C and D are concepts, R a role, a and b are individuals
- An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$
- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

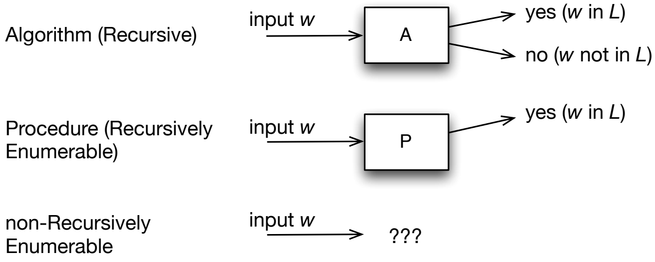
- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$
- A knowledge base $\mathcal{KB}$ is said to be satisfiable if it admits a model

Outline

1. Introduction
2. Basic DL: $\mathcal{ALC}$
   - Syntax
   - Semantics
3. DL and FOL
4. Reasoning services
   - Standard services
   - Techniques

DLs are structured fragments of FOL

- Recall that full FOL is **undecidable**
- This is unpleasant for automated reasoning



DLs are structured fragments of FOL

- Approach: find a fragment—a *sublanguage*—of FOL that is decidable
- Take some features, prove the computational complexity of some problem
- But lest first demonstrate the two are related, so that we can do this

Example correspondences

- $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$

Example correspondences

- $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$
- $C \sqsubseteq D \sqcap E$
  - $\forall x(C(x) \rightarrow D(x) \wedge E(x))$

Example correspondences

- $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$
- $C \sqsubseteq D \sqcap E$
  - $\forall x(C(x) \rightarrow D(x) \wedge E(x))$
- $C \sqsubseteq \exists R.D$
  - $\forall x(C(x) \rightarrow \exists y(R(x, y) \wedge D(y))$

Example correspondences

- $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$
- $C \sqsubseteq D \sqcap E$
  - $\forall x(C(x) \rightarrow D(x) \wedge E(x))$
- $C \sqsubseteq \exists R.D$
  - $\forall x(C(x) \rightarrow \exists y(R(x,y) \wedge D(y))$
- $C \equiv \exists R.D \sqcup \exists S.D$
  - $\forall x(C(x) \leftrightarrow \exists y((R(x,y) \vee S(x,y)) \wedge D(y))$

DLs are structured fragments of FOL

- We end up with *trade-offs* of features in a DL
- Some features always will make the language undecidable (e.g., true role composition, $R \circ S \equiv T$)
- Other features are only 'problematic' (computationally less desirable) when taken together with another

DLs are structured fragments of FOL

- We end up with *trade-offs* of features in a DL
- Some features always will make the language undecidable (e.g., true role composition, $R \circ S \equiv T$)
- Other features are only 'problematic' (computationally less desirable) when taken together with another
- E.g., one could define a language where:
  - it is prohibited to use $\forall$ in an axiom, or
  - only $\exists R.\top$ (no range specified) but not $\exists R.D$, or
  - $\exists R$ only on the rhs of the inclusion but not on the lhs

DLs are structured fragments of FOL

- We end up with *trade-offs* of features in a DL
- Some features always will make the language undecidable (e.g., true role composition, $R \circ S \equiv T$)
- Other features are only 'problematic' (computationally less desirable) when taken together with another
- E.g., one could define a language where:
  - it is prohibited to use $\forall$ in an axiom, or
  - only $\exists R.\top$ (no range specified) but not $\exists R.D$, or
  - $\exists R$ only on the rhs of the inclusion but not on the lhs
- There are *many* DLs, and most combinations have been investigated over the past 25 years
- Roughly: the fewer features and the more restrictions, the more 'computationally well-behaved' the language is

Outline

Essential to automated reasoning

- The choice of the class of problems the software program has to solve
- The formal language in which to represent the problems
- The way how the program has to compute the solution
- How to do this efficiently

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists \mathtt{TEACHES}.\mathtt{Course} \sqsubseteq \neg \mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES}(\mathtt{John}, \mathtt{cs101})$, $\mathtt{Course}(\mathtt{cs101})$, $\mathtt{Undergrad}(\mathtt{John})$

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists\mathtt{TEACHES}.\mathtt{Course} \sqsubseteq \neg\mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES}(\mathtt{John}, \mathtt{cs101})$, $\mathtt{Course}(\mathtt{cs101})$, $\mathtt{Undergrad}(\mathtt{John})$
- $\mathcal{KB} \models \mathtt{Professor}(\mathtt{John})$

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists \mathtt{TEACHES.Course} \sqsubseteq \neg \mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES(John, cs101)}$, $\mathtt{Course(cs101)}$, $\mathtt{Undergrad(John)}$
- $\mathcal{KB} \models \mathtt{Professor(John)}$
- What if:
  TBox: $\exists \mathtt{TEACHES.Course} \sqsubseteq \mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES(John, cs101)}$, $\mathtt{Course(cs101)}$, $\mathtt{Undergrad(John)}$
- $\mathcal{KB} \models \mathtt{Professor(John)}$? or perhaps
  $\mathcal{KB} \models \neg\mathtt{Professor(John)}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?
- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?
- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of C in $\mathcal{KB}$, i.e., compute all individuals a s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$

Automated reasoning techniques

- How do we compute, say, satisfiability?

Automated reasoning techniques

- How do we compute, say, satisfiability?
- Truth tables are too cumbersome
- Several techniques are more efficient
- Current 'winner' is *tableau reasoning*

The idea–same as for FOL

- A sound and complete procedure deciding satisfiability is all we need, and the **tableaux method is a decision procedure which checks the existence of a model**
- It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.
- $\phi \models \psi$ iff $\phi \land \neg\psi$ is NOT satisfiable—if it is satisfiable, we have found a counterexample
- Decompose the formula in top-down fashion

Basic rules (from previous lecture)

- Tableaux calculus works only if the formula has been translated into Negation Normal Form, *i.e.*, all the negations have been pushed inside
- If a model satisfies a conjunction, then it also satisfies each of the conjuncts
- If a model satisfies a disjunction, then it also satisfies one of the disjuncts. It is a non-deterministic rule, and it generates two alternative branches.
- Apply the completion rules until either (a) an explicit contradiction due to the presence of two opposite literals in a node (a clash) is generated in each branch, or (b) there is a completed branch where no more rule is applicable.

Example (from previous lecture)

Number	Tableau	Explanation
1	$\forall x.R(x,x)$	Reflexivity axiom in the original theory T
2	$\forall x,y.\ \neg R(x,y) \vee \neg R(y,x)$	Asymmetry axiom in the original theory T
3	$\forall x,y.R(x,y)$	The negated axiom added to theory T
4		Substitute *x* for term *a* in 1,2,3
5	$R(a,a)$	
6	$\forall y.\ \neg R(a,y) \vee \neg R(y,a)$	
7	$\forall y.R(a,y)$	
8		Substitute *y* for term *a* in 2 and 3
9	$R(a,a)$	
10	$\neg R(a,a) \vee \neg R(a,a)$	
11	$R(a,a)$	
12		Split the disjunction of 10
13	$\neg R(a,a)$ $\quad$ $\neg R(a,a)$	Which each generate a clash with 9 and 11, hence, $\neg\forall x,y.R(x,y)$ is entailed by T.

Tableau reasoning for DLs

- Most common for DL reasoners
- Like for FOL:
  - Unfold the TBox
  - Convert the result into negation normal form
  - Apply the tableau rules to generate more Aboxes
  - Stop when none of the rules are applicable
- $\mathcal{T} \vdash C \sqsubseteq D$ if all Aboxes contain clashes
- $\mathcal{T} \nvdash C \sqsubseteq D$ if some Abox does not contain a clash

A note on soundness and completeness

- "$\vdash$": derivable with a set of inference rules,
- "$\models$" as implies, i.e., every truth assignment that satisfies Γ also satisfies ϕ

A note on soundness and completeness

- "$\vdash$": derivable with a set of inference rules,
- "$\models$" as implies, i.e., every truth assignment that satisfies Γ also satisfies ϕ
- Completeness: if $\Gamma \models \phi$ then $\Gamma \vdash \phi$
  - If the algorithm is *incomplete*, then there exist entailments that cannot be computed (hence, 'missing' some results)

A note on soundness and completeness

- "$\vdash$": derivable with a set of inference rules,
- "$\models$" as implies, i.e., every truth assignment that satisfies Γ also satisfies ϕ
- Completeness: if $\Gamma \models \phi$ then $\Gamma \vdash \phi$
  - If the algorithm is *incomplete*, then there exist entailments that cannot be computed (hence, 'missing' some results)
- Soundness: if $\Gamma \vdash \phi$ then $\Gamma \models \phi$
  - If the algorithm is *unsound* then false conclusions can be derived from true premises, which his even more undesirable

Negation Normal Form

- C and D are concepts, R a role
- $\neg$ only in front of concepts:
  - $\neg\neg C$ gives C
  - $\neg(C \sqcap D)$ gives $\neg C \sqcup \neg D$
  - $\neg(C \sqcup D)$ gives $\neg C \sqcap \neg D$
  - $\neg(\forall R.C)$ gives $\exists R.\neg C$
  - $\neg(\exists R.C)$ gives $\forall R.\neg C$

Tableau rules for $\mathcal{ALC}$

$\sqcap$-rule If $(C_1 \sqcap C_2)(a) \in S$ but S does not contain both $C_1(a)$ and $C_2(a)$, then
$S = S \cup \{C_1(a), C_2(a)\}$

$\sqcup$-rule If $(C_1 \sqcup C_2)(a) \in S$ but S contains neither $C_1(a)$ nor $C_2(a)$, then
$S = S \cup \{C_1(a)\}$
$S = S \cup \{C_2(a)\}$

$\forall$-rule If $(\forall R.C)(a) \in S$ and S contains $R(a, b)$ but not $C(b)$, then
$S = S \cup \{C(b)\}$

$\exists$-rule If $(\exists R.C)(a) \in S$ and there is no b such that $C(b)$ and $R(a, b)$, then
$S = S \cup \{C(b), R(a, b)\}$

Example

- Let's say our ontology contains only:
  - 1a $Vegan \equiv Person \sqcap \forall eats.Plant$
  - 1b $Vegetarian \equiv Person \sqcap \forall eats.(Plant \sqcup Dairy)$
- We want to know whether all vegans are vegetarians, i.e.: $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$

Example

- Let's say our ontology contains only:
  - 1a $Vegan \equiv Person \sqcap \forall eats.Plant$
  - 1b $Vegetarian \equiv Person \sqcap \forall eats.(Plant \sqcup Dairy)$
- We want to know whether all vegans are vegetarians, i.e.: $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$
- If that's true, then there is, or can be, an individual that is an instance of both, or:
- If that's true, then some object that instantiates the subclass but *not* the superclass *cannot* exist
  - 2 $\mathcal{S} = \{(Vegan \sqcap \neg Vegetarian)(a)\}$

Example

- Let's say our ontology contains only:
  - 1a $Vegan \equiv Person \sqcap \forall eats.Plant$
  - 1b $Vegetarian \equiv Person \sqcap \forall eats.(Plant \sqcup Dairy)$
- We want to know whether all vegans are vegetarians, i.e.: $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$
- If that's true, then there is, or can be, an individual that is an instance of both, or:
- If that's true, then some object that instantiates the subclass but *not* the superclass *cannot* exist
  - 2 $\mathcal{S} = \{(Vegan \sqcap \neg Vegetarian)(a)\}$
- Before entering the tableau, we'll 'unfold' it (informally, here: complex concepts on the left-hand side are replaced with their properties declared on the right-hand side)

Example

- Let's say our ontology contains only:
  - 1a $Vegan \equiv Person \sqcap \forall eats.Plant$
  - 1b $Vegetarian \equiv Person \sqcap \forall eats.(Plant \sqcup Dairy)$
- We want to know whether all vegans are vegetarians, i.e.: $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$
- If that's true, then there is, or can be, an individual that is an instance of both, or:
- If that's true, then some object that instantiates the subclass but *not* the superclass *cannot* exist
  - 2 $\mathcal{S} = \{(Vegan \sqcap \neg Vegetarian)(a)\}$
- Before entering the tableau, we'll 'unfold' it (informally, here: complex concepts on the left-hand side are replaced with their properties declared on the right-hand side)
- Check for NNF and rewrite if needed
- Then (finally) apply the tableau rules

Summary

1. Introduction
2. Basic DL: $\mathcal{ALC}$
   - Syntax
   - Semantics
3. DL and FOL
4. Reasoning services
   - Standard services
   - Techniques