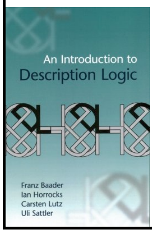
THE UNIVERSITY *of* EDINBURGH

INFR11215 Knowledge Graphs

DL Reasoning with Tableaux Algorithms

Jeff Z. Pan

http://knowledge-representation.org/j.z.pan/

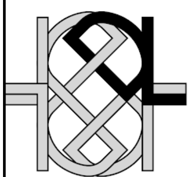


[Reading: Baader et al., sections 2.3, 4.1 and 4.2]

1

THE UNIVERSITY *of* EDINBURGH

ALC Knowledge Base



Let **C** and **R** be disjoint sets of concept names and role names, respectively.

$\mathcal{ALC}$-concept descriptions are defined by induction:

- If $A \in \mathbf{C}$, then A is an $\mathcal{ALC}$-concept description.
- If C, D are $\mathcal{ALC}$-concept descriptions, and $r \in \mathbf{R}$, then the following are $\mathcal{ALC}$-concept descriptions:
  - $C \sqcap D$ (conjunction)
  - $C \sqcup D$ (disjunction)
  - $\neg C$ (negation)
  - $\forall r.C$ (value restriction)
  - $\exists r.C$ (existential restriction)

Abbreviations:
- $\top := A \sqcup \neg A$ (top)
- $\bot := A \sqcap \neg A$ (bottom)
- $C \Rightarrow D := \neg C \sqcup D$ (implication)

$\mathcal{ALC}$ Knowledge Base K=(T,A), where T is anTBox (containing only class subsumption axioms C $\sqsubseteq$ D only) and A is an ABox

[credit: F Baader]

2

Class Axioms in ALC (1)

- SubClassOf axioms
  - DL syntax: $C1 \sqsubseteq C2$
  - FOL syntax: $\forall x\ [C1(x) \rightarrow C2(x)]$
- Equivalent Class axioms
  - DL syntax: $C1 \equiv C2$
  - Or, $C1 \sqsubseteq C2$, $C2 \sqsubseteq C1$

ALC Class Axioms (2)

- They are also called axioms, or schema axioms
- Disjoint Class axioms
  - DL syntax: $C1 \sqsubseteq \neg C2$
- Exhaustive Class axioms
  - DL syntax: $C \sqsubseteq C1 \sqcup C2$

THE UNIVERSITY of EDINBURGH

ALC Property Axioms

- Property Domain axioms
  - DL syntax: $\exists r \sqsubseteq C$
  - FOL syntax: $\forall x[\exists y.r(x,y) \rightarrow C(x)]$
- Property Range axioms
  - DL syntax: ~~$\exists r^{-} \sqsubseteq C$~~
  - DL Syntax: $\top \sqsubseteq \forall r.C$
  - FOL syntax: $\forall x[\exists y.r(y,x) \rightarrow C(x)]$

THE UNIVERSITY of EDINBURGH

ALC Assertions

- Class Assertions
  - DL syntax: e:A, or A(e)
  - RDF Notation 3 (N3) syntax: [e rdf:type A .]
- Property Assertions
  - DL syntax: (e1,e2):r, or r(e1, e2)
  - RDF N3 syntax: [e1 r e2 .]
- Equality / Inequality assertions
  - e1 = e2
  - e1 ≠ e2

THE UNIVERSITY of EDINBURGH

DL Interpretations

- An interpretation **I** is written as $(\Delta^I, \bullet^I)$
  - Δ^I is the **non-empty domain**
  - $\bullet^I$ is the **interpretation function**
    - all individuals (inc. unnamed ones) are members of the domain: $o^I \in \Delta^I$
    - all classes are subsets of the domain $A^I \subseteq \Delta^I$
      - e.g., $Employee^I$= {E1, E2, E3, E4}
    - all properties are subsets $R^I \subseteq \Delta^I \times \Delta^I$

      e.g., $Works\text{-}for^I$= {<E1,P1>, <E2,P1>, <E2,P2>, <E3,P1>, <E3,P2>, <E4,P2>}
- Interpretation function allows us to consider all possible **assignment of class and property memberships**
  - all possible databases for the given schema

Knowledge Graphs
Jeff Z. Pan

7

7

THE UNIVERSITY of EDINBURGH

DL Interpretations

- KG schema (Ontology)
  - President $\sqsubseteq$ Politician
- Question: does the following interpretation satisfy the above axiom?
  - Δ^I = {Obama, Trump, Biden}
  - $President^I$ = {Obama, Trump, Biden}
  - $Politiciant^I$ = {Obama, Biden}

Knowledge Graphs
Jeff Z. Pan

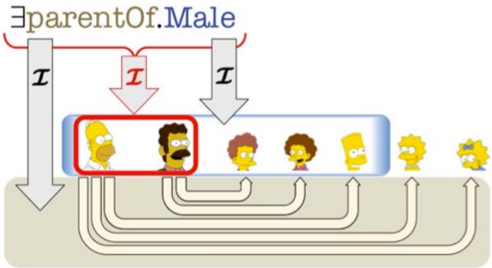
8

8

Interpretations of Restrictions

Given an interpretation, we can compute the semantic counterparts of class descriptions

$$\exists r.C = \{ x \mid \exists y.\ (x,y) \in r^{\mathcal{I}} \land y \in C^{\mathcal{I}} \}$$
$$\forall r.C = \{ x \mid \forall y.\ (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}} \}$$

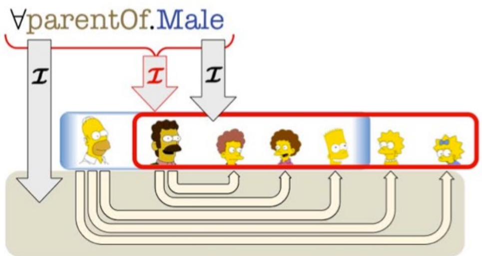


THE UNIVERSITY of EDINBURGH

Interpretations of Restrictions

Given an interpretation, we can compute the semantic counterparts of class descriptions

$$\exists r.C = \{ x \mid \exists y.\ (x,y) \in r^{\mathcal{I}} \land y \in C^{\mathcal{I}} \}$$
$$\forall r.C = \{ x \mid \forall y.\ (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}} \}$$



THE UNIVERSITY of EDINBURGH

Lecture Outline

- Motivation
- Overview of Tableau Algorithms
- More details on Tableau Algorithms
- Practical

THE UNIVERSITY of EDINBURGH

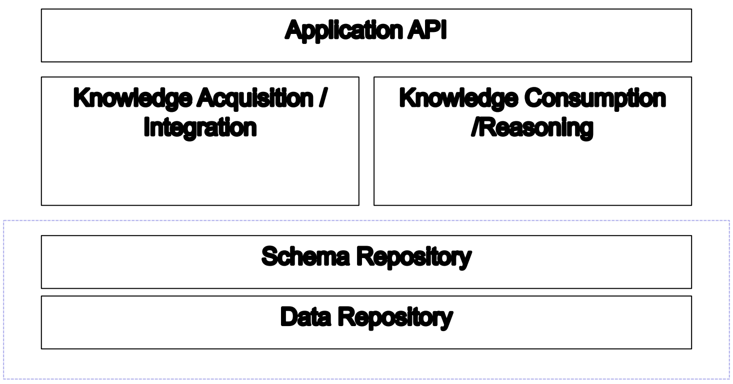
Motivations

- It is not an easy task to come up with decision procedures for reasoning services even for simpler DLs
  - Some early algorithms are incomplete
- One stone few birds
  - One algorithm for four reasoning services

Lecture Outline

- Motivation
- Overview of Tableau Algorithms
- More details on Tableau Algorithms
- Practical

Architecture of Knowledge Based Systems



THE UNIVERSITY of EDINBURGH

Ontology and Reasoning

- Ontology contains
  - knowledge and data that
  - we know that we know
  - we know that we don't know or partially know
- Reasoning helps to find out
  - things that we might not know that we know
- Combining reasoning and learning
  - things that we might not know that we don't know

THE UNIVERSITY of EDINBURGH

DL Reasoning Services

- KB consistency checking
  - An KB is consistent, if there exist an interpretation that satisfies all axioms in KB
- Class satisfiability checking
  - A class description C is satisfiable w.r.t. a KB, if there exist an interpretation (model) I of KB, such as C^I is non-empty
- Class subsumption checking
  - C is subsumed by D satisfiable w.r.t. a KB, if in all interpretations (models) I of KB, C^I is subfset of D^I
- Instance Checking
  - KB infers C(a) [r(a,b)] if, in all interpretations (models) of KB, a $\in C^I$ [$(a^I, b^I) \in r^I$]
- All reducible to KB consistency checking

THE UNIVERSITY of EDINBURGH

Tableaux Algorithm

- The first sound and complete algorithm for expressive DLs
  - **Ontology Consistency Checking**
- Basic idea: Build an interpretation
  - A **tableau** is a representative of an interpretation
    - Δ^I is the **non-empty domain**
  - We can construct an interpretation based on a tableau

Knowledge Graphs
Jeff Z. Pan

17

17

THE UNIVERSITY of EDINBURGH

Tableaux Algorithm: Key Steps

1. Initialise the tableau with individual axioms
   - the initial tableau might not satisfy all the axioms
2. Repair the initial tableau by applying expansion rules
   - so as to add new information into the tableau
   - this might require backtracking
3. If the tableau satisfy all the axioms, returns Consistent
4. If every possible attempt repair results in some contradiction, returns Inconsistent
   - Contradiction: o: A, o: ¬A, or o: ⊥ in the expanded ABox, (⊥ is bottom, interpreted as **empty set**)

Knowledge Graphs
Jeff Z. Pan

18

18

NNF: Negated Normal Form

- Negated Normal Form (NNF)
  - If a class is in NNF, negations only appear in front of named classes
  - E.g., ¬Person is in NNF
  - but ¬(Chinese ⊓ English) is not in NNF
- In tableau algorithm, all the input classes should be in NNF
  - We can make use of the following table to transform inputs into NNF

$$\neg\neg C \equiv C \qquad \neg\exists r.C \equiv \forall r.\neg C$$
$$\neg(C \sqcap D) \equiv \neg D \sqcup \neg C \qquad \neg\forall r.C \equiv \exists r.\neg C$$
$$\neg(C \sqcup D) \equiv \neg D \sqcap \neg C \qquad \neg\leqslant nr.C \equiv \geqslant(n+1)r.C$$
$$\neg\geqslant(n+1)r.C \equiv \leqslant nr.C$$

19

Expansion Rules

The ⊓-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcap D)$, but not both $a:C$ and $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:C, a:D\}$

The ⊔-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcup D)$, but neither $a:C$ nor $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:X\}$ for some $X \in \{C, D\}$

The ∃-rule

Condition: $\mathcal{A}$ contains $a:(\exists r.C)$, but there is no b with $\{(a,b):r, b:C\} \subseteq \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d):r, d:C\}$ where d is new in $\mathcal{A}$

The ∀-rule

Condition: $\mathcal{A}$ contains $a:(\forall r.C)$ and $(a,b):r$, but not $b:C$

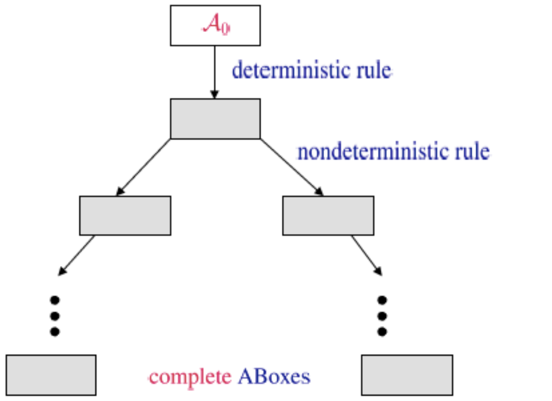
Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{b:C\}$

[credit: F Baader]

20

THE UNIVERSITY of EDINBURGH

⊔-rule: A Non-deterministic Rule for backtracking



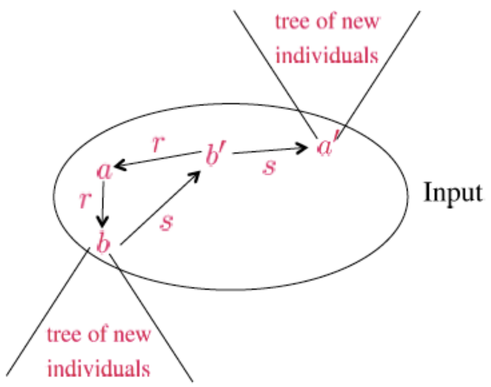
Return "consistent" iff one of these complete ABoxes is clash-free.

[credit: F Baader]

THE UNIVERSITY of EDINBURGH

∃-rule Create New Individuals

- In the expanded ABox, new individual created by the ∃-rule form trees, whose roots are in the original ABox



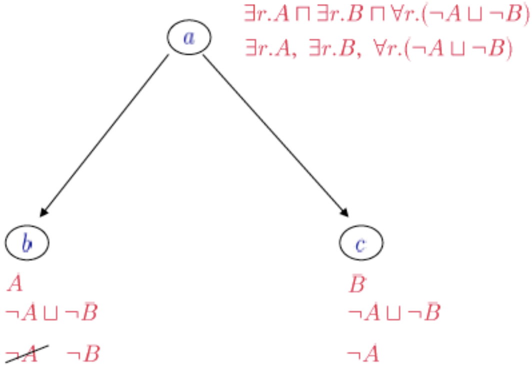
[credit: F Baader]

THE UNIVERSITY of EDINBURGH

Tableaux Algorithm: Example (without TBox)

Is the following ABox inconsistent?

{ a :(∃attended.Smart ⊓ ∃attended.Studious ⊓ ∀attended.(¬Smart ⊔ ¬Studious)) }



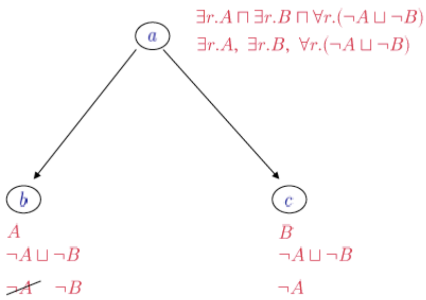
[credit: F Baader]

THE UNIVERSITY of EDINBURGH

Tableaux Algorithm: Example (without TBox)

Is the following ABox inconsistent?

{ a :(∃attended.Smart ⊓ ∃attended.Studious ⊓ ∀attended.(¬Smart ⊔ ¬Studious)) }



- Δ^I={a,b,c}
- Smart^I = {b}, Studious^I = {c}
- attended^I = {<a,b>,<a,c>}

THE UNIVERSITY of EDINBURGH

Lecture Outline

- Motivation
- Overview of Tableau Algorithms
- More details on Tableau Algorithms
- Practical

Knowledge Graphs
Jeff Z. Pan

25

25

THE UNIVERSITY of EDINBURGH

S-rule for Simple Axioms

- Simple axioms
  - A⊑C where A is a named class
  - No cycles involve A
    - ✘ such as A ⊑∃R.A
- **Expansion rule for simple axioms**
  - If x:A is ABox and A ⊑ C is in TBox
  - Then add x:C into ABox

Knowledge Graphs
Jeff Z. Pan

26

26

THE UNIVERSITY of EDINBURGH

Example: S-rule

- Check if the following ontology is consistent
  1. English ⊑ ¬Chinese
  2. Confucian ⊑Chinese
  3. Confucian ⊑English
  4. Bill : Confucian

5. Initialise the tableau; A={Bill: Confucion} (from 4)
6. Apply S-rule on 2 and 3, A={Bill: Confucion, Bill: Chinese, Bill: English}
7. Apply S-rule on 1, A={Bill: Confucion, Bill: Chinese, Bill: English, Bill: ¬Chinese}.
8. Since there is a clash and no backtrack, the ontology is inconsistent

THE UNIVERSITY of EDINBURGH

Expansion Rules for Acyclic TBoxes

- One step beyond simple axioms
  - A ≡ C (the same as A ⊑C, C ⊑A)

S-rule

The $\equiv_1$-rule

Condition: $a : A \in \mathcal{A}$, $A \equiv C \in \mathcal{T}$, and $a : C \notin \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C\}$

S⁻-rule

The $\equiv_2$-rule

Condition: $a : \neg A \in \mathcal{A}$, $A \equiv C \in \mathcal{T}$, and $a : \dot{\neg} C \notin \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : \dot{\neg} C\}$

$\dot{\neg} C$
Negation normal form of $\neg C$

[credit: F Baader]

THE UNIVERSITY of EDINBURGH

Expansion Rules for GCI

- General Class Inclusion (GCI)
  - C $\sqsubseteq$ D, where C is a class desription and not a named class
  - Idea: turn the left handside into $\top$ (top), thus it is applicable to every individual in the tableau
  - How?
    - Since C $\sqcup \neg$C is equivalent to $\top$, we can turn C$\sqsubseteq$ D into $\top \sqsubseteq$D $\sqcup \neg$C
- GCI is expensive to deal with
  - since it adds a disjunctive to every individual
- Don't forget to use NNF

THE UNIVERSITY of EDINBURGH

Class Satisfiability Checking

- Class satisfiability checking can be reduced to ontology consistency checking
  - by assuming the target class C has an instance x
  - hence adding x:C into O: O'=O U {x:C}
  - If O' is inconsistent, then the assumption is invalid, so C is insatisfiable
  - Otherwise, C is satisfiable

THE UNIVERSITY of EDINBURGH

Blocking: Ensuring Termination

- Expansion can be applicable forever
  - We need to block the expansion on e.g. cyclic axioms
- Blocking
  - Let Sub(x) be the subset of A that includes all class asserstions about x
  - Condition: Sub(y) $\subseteq$ Sub(x) for some ancestor x (blocking node) and predecessor y (blocked node)
  - Intuitively, this means that the same constraints have been dealt with before

Knowledge Graphs
Jeff Z. Pan

31

31

THE UNIVERSITY of EDINBURGH

Example: Blocking

- Example:
  - Given the ontology
    1. Person ⊑∃friend.Person
  - Check if Person is satisfiable
- Construct a tableau
  2. Initialise the ABox A={x:Person}
  3. Apply S-rule on 1, A={x:Person, x:∃friend.Person}
  4. Apply ∃-rule, A={x:Person, x:∃friend.Person, (x,x1):friend, x1:Person}
  5. Sub(x1) $\subseteq$ Sub(x), so x1 is blocked and replaced by x, A={x:Person, x:∃friend.Person, (x,x):friend }
  6. We can construct an interpretation as follows:
     - Δ^I={x}
     - $Person^I$ = {x}
     - $friend^I$ = {<x,x>}

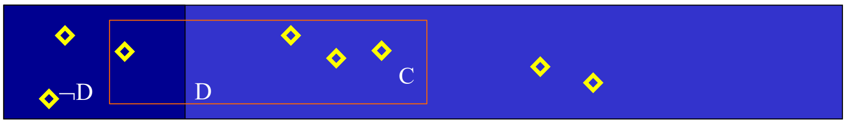
Knowledge Graphs
Jeff Z. Pan

32

32

Class Subsumption Checking

- Class satisfiability checking can be reduced to class (un)satisfiability checking
  - $O \models C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable



Example: Class Subsumption Checking

- Check if the following subsumption holds
  - $A \sqcap \exists r.A \sqcap \forall r.B \sqsubseteq A \sqcap \exists r.B$

1. To test the subsumption, we need to check the satisfiability of the class $A \sqcap \exists r.A \sqcap \forall r.B \sqcap \neg(A \sqcap \exists r.B)$
2. Turn it into NNF: $A \sqcap \exists r.A \sqcap \forall r.B \sqcap (\neg A \sqcup \forall r.\neg B)$
3. Initial the ABox: $A1=\{x: A \sqcap \exists r.A \sqcap \forall r.B \sqcap (\neg A \sqcup \forall r.\neg B)\}$
4. Apply $\sqcap$-rule: $A2=A1 \cup \{x:A, x: \exists r.A, x: \forall r.B, x: \neg A \sqcup \forall r.\neg B\}$
5. Apply $\exists$-rule: $A3 = A2 \cup \{\langle x,x1\rangle:r, x1:A\}$
6. Apply $\forall$-rule: $A4=A3 \cup \{x1:B\}$
7. Apply $\sqcup$-rule: $A5=A4 \cup \{x: \neg A\}$, clash
8. Backtrack, $A5'= A4 \cup \{x: \forall r.\neg B\}$
9. Apply $\forall$-rule: $A6 = A5' \cup \{x1: \neg B\}$, clash, not backtractable
10. The test class description is unsatisfiable, so the subsumption holds

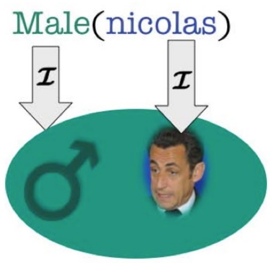
Class Instance Checking

- Reducing Class Instance Checking to Ontology Consistency Checking
  - If O entails C(x), then in every interpretation I of O, we have x^I is in C^I
  - It means O U {¬C(x)} inconsistent

THE UNIVERSITY of EDINBURGH

Example

- If an ontology O entails Male (nicolas)
  - then in every interpretation I of O
  - we have $nicolas^I \in Male^I$
- Now if we extend O to O' with a new axiom
  - ¬Male(nicolas) (*)
- How to construct an interpretation I' for O'?
  - as all interpretations of O' should satisfy O
  - we could start from interpretations of O
- It is easy to see that I' does not exist
  - If I' does not satisfy O, then it does not satisfy O' either
  - If I' satisfies O, then it does not satisfy ¬Male(nicolas)



THE UNIVERSITY of EDINBURGH

Class Instance Checking

- Question: given the following ontology O,
  - OldLady ⊑ ∀ hasPet.Cat
  - OldLady(Minnie)
  - hasPet(Minnie, Tom)
- Does O entail Tom: Cat?

1. Add Tom: ¬ Cat into the ontology O
2. Initial the tableau: A={Minnie:OldLady, <Minnie,Tom>:hasPet,Tom: ¬ Cat}
3. Apply S-rule on axiom 1 with Minnie: A1=A U {Minnie: ∀hasPet.Cat}
4. Apply ∀-rule on Minnie: A2=A1 U {Tom:Cat} clash, not backtractable
5. Thus the extended ontology is inconsistent
6. And the entailment holds

Knowledge Graphs
Jeff Z. Pan 37

37

THE UNIVERSITY of EDINBURGH

Example: Consistency Checking

$$\mathcal{T} := \{\neg(A \sqcup B) \sqsubseteq \bot, \quad A \sqsubseteq \neg B \sqcap \exists r.B, \quad D \sqsubseteq \forall r.A, \quad B \sqsubseteq \neg A \sqcap \exists r.A\}$$

1. Rewrite the first axiom into ⊤ ⊑ A ⊔ B
2. Since any individual (such as x) is an instance of ⊤, x must be an instance of A ⊔ B
3. Initialise the tableau: A1={x: A⊔B}
4. Apply ⊔-rule on x: A2=A1 U {x:A}
5. Apply S-rule on axiom2 with x: A3=A2 U {x: ¬B ⊓∃r.B}
6. Apply ⊓-rule on x: A4=A3 U {x: ¬B, x: ∃r.B}
7. Apply ∃-rule on x: A5=A4 U {<x,x1>:r, x1:B}
8. Apply GCI-rule on axiom 1 with x1: A6=A5 U {x1: A ⊔B}
9. Apply ⊔-rule on x1: A7=A6 U {x1:B}
10. Apply S-rule on axiom 4 with x1: A8=A7 U {¬A ⊓ ∃r.A}
11. Apply ⊓-rule on x1: A9=A8 U {¬A, ∃r.A}
12. Apply ∃-rule on x1: A10=A9 U {<x1,x2>:r, x2:A}
13. x2 is blocked by x
14. All axioms have been dealt with, and there is no contradiction, so that the TBox is consistent

Knowledge Graphs
Jeff Z. Pan 38

38

THE UNIVERSITY of EDINBURGH

Lecture Outline

- Motivation: Sound and complete reasoning in DL
- Introduction: Tableau algorithm
- Focus: The ALC DL
- Exercises (Mid-term Quiz next week)
  - Check the consistency of the following knowledge graph:

$$\begin{aligned} \mathcal{A}_{ex} = \{ a : A \sqcap \exists s.F, \quad & (a,b) : s, \\ a : \forall s.(\neg F \sqcup \neg B), \quad & (a,c) : r, \\ b : B, \quad & c : C \sqcap \exists s.D \}. \end{aligned}$$