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]

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

Abbreviations:
- \( T \) := \( A \lor \neg A \) (top)
- \( \bot \) := \( A \land \neg A \) (bottom)
- \( C \Rightarrow D \) := \( \neg C \lor D \) (implication)

\( \mathcal{ALC} \) Knowledge Base \( K = (T, A) \), where \( T \) is an TBox (containing only class subsumption axioms \( C \sqsubseteq D \) only) and \( A \) is an ABox
Class Axioms in ALC (1)

- **SubClassOf axioms**
  - DL syntax: \( C_1 \sqsubseteq C_2 \)
  - FOL syntax: \( \forall x [C_1(x) \rightarrow C_2(x)] \)
- **Equivalent Class axioms**
  - DL syntax: \( C_1 \equiv C_2 \)
  - Or, \( C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1 \)

ALC Class Axioms (2)

- They are also called axioms, or schema axioms
- **Disjoint Class axioms**
  - DL syntax: \( C_1 \sqsubseteq \neg C_2 \)
- **Exhaustive Class axioms**
  - DL syntax: \( C \sqsubseteq C_1 \sqcup C_2 \)
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: $r \sqsubseteq C$
  – DL Syntax: $\top \sqsubseteq \forall r.C$
  – FOL syntax: $\forall x[\exists y. r(y,x) \rightarrow C(x)]$

ALC Assertions

• Class Assertions
  – DL syntax: $e:A$, or $A(e)$
  – RDF Notation 3 (N3) syntax: $[e$ rdf:type $A]$ .
• Property Assertions
  – DL syntax: $(e_1,e_2):r$, or $r(e_1, e_2)$
  – RDF N3 syntax: $[e_1$ r $e_2]$ .
• Equality / Inequality assertions
  – $e_1 = e_2$
  – $e_1 \neq e_2$
DL Interpretations

• An interpretation $I$ is written as $(\Delta^I, \cdot^I)$
  – $\Delta^I$ is the **non-empty domain**
  – $\cdot^I$ is the **interpretation function**
    • all individuals (inc. unnamed ones) are members of the domain: $\circ^I \in \Delta^I$
    • 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-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

DL Interpretations

• KG schema (Ontology)
  – President $\sqsubseteq$ Politician

• Question: does the following interpretation satisfy the above axiom?
  – $\Delta^I = \{Obama, Trump, Biden\}$
  – President$^I = \{Obama, Trump, Biden\}$
  – Politician$^I = \{Obama, Biden\}$
Interpretations of Restrictions

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

\[ \exists r \cdot C = \{ x \mid \exists y. (x, y) \in r^I \land y \in C^I \} \]
\[ \forall r \cdot C = \{ x \mid \forall y. (x, y) \in r^I \rightarrow y \in C^I \} \]
**Lecture Outline**

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

---

**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

- Application API
- Knowledge Acquisition / Integration
- Knowledge Consumption / Reasoning
- Schema Repository
- Data Repository
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

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, $a \in C^I [ (a', b') \in r^I ]$
- All reducible to KB consistency checking
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
    • $\Delta'$ is the non-empty domain
  – We can construct an interpretation based on a tableau

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: $\circ: A$, $\circ: \neg A$, or $\circ: \bot$ in the expanded ABox, ($\bot$ is bottom, interpreted as empty set)
**NNF: Negated Normal Form**

- **Negated Normal Form (NNF)**
  - If a class is in NNF, negations only appear in front of named classes
  - E.g., \( \neg \)Person is in NNF
  - but \( \neg (\text{Chinese} \cap \text{English}) \) 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

\[
\begin{align*}
\neg \neg C & \equiv C \\
\neg (C \cap D) & \equiv \neg D \cup \neg C \\
\neg (C \cup D) & \equiv \neg D \cap \neg C \\
\neg \exists r.C & \equiv \forall r.\neg C \\
\neg \forall r.C & \equiv \exists r.\neg C \\
\neg (n+1) r. C & \equiv \forall r. C \\
(n+1) r. C & \equiv \exists r. C \\
\end{align*}
\]

---

**Expansion Rules**

- **The \( \cap \)-rule**
  - **Condition**: \( \mathcal{A} \) contains \( a : (C \cap D) \), but not both \( a : C \) and \( a : D \)
  - **Action**: \( \mathcal{A} \rightarrow \mathcal{A} \cup \{a : C, a : D\} \)

- **The \( \cup \)-rule**
  - **Condition**: \( \mathcal{A} \) contains \( a : (C \cup D) \), but neither \( a : C \) nor \( a : D \)
  - **Action**: \( \mathcal{A} \rightarrow \mathcal{A} \cup \{a : X\} \) for some \( X \in \{C, D\} \)

- **The \( \exists \)-rule**
  - **Condition**: \( \mathcal{A} \) contains \( a : (\exists r. C) \), but there is no \( b \) with \( \{(a, b) : r, b : C\} \subseteq \mathcal{A} \)
  - **Action**: \( \mathcal{A} \rightarrow \mathcal{A} \cup \{(a, d) : r, d : C\} \) where \( d \) is new in \( \mathcal{A} \)

- **The \( \forall \)-rule**
  - **Condition**: \( \mathcal{A} \) contains \( a : (\forall r. C) \) and \( (a, b) : r \), but not \( b : C \)
  - **Action**: \( \mathcal{A} \rightarrow \mathcal{A} \cup \{b : C\} \)

[credit: F. Baader]
\(-\text{rule: A Non-deterministic Rule for backtracking}\)

\[\mathcal{A}_0\]

deterministic rule

\[\text{nondeterministic rule}\]

\[\text{complete ABoxes}\]

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

[credit: F Baader]

---

\(\exists\)-rule Create New Individuals

- In the expanded ABox, new individual created by the \(\exists\)-rule form trees, whose roots are in the original ABox

[credit: F Baader]
Tableaux Algorithm: Example (without TBox)

Is the following ABox inconsistent?

\{ a : (\exists \text{attended. Smart} \land \exists \text{attended. Studious} \land \forall r. (\neg \text{Smart} \lor \neg \text{Studious})) \}

\exists r. A \land \exists r. B \land \forall r. (\neg A \lor \neg B)

\exists r. A, \exists r. B, \forall r. (\neg A \lor \neg B)

- $A = \{a, b, c\}$
- $\text{Smart}^d = \{b\}$, $\text{Studious}^d = \{c\}$
- $\text{attended}^d = \{<a, b>, <a, c>\}$
Lecture Outline

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

S-rule for Simple Axioms

- Simple axioms
  - $\text{A}\sqsubseteq\text{C}$ where A is a named class
  - No cycles involve A
    - $\times$ such as $\Box\text{R.A}$
- Expansion rule for simple axioms
  - If $x:A$ is ABox and $A\sqsubseteq C$ is in TBox
  - Then add $x:C$ into ABox
Example: S-rule

- Check if the following ontology is consistent
  1. English $\sqsubseteq \neg$ Chinese
  2. Confucian $\sqsubseteq$ Chinese
  3. Confucian $\sqsupseteq$ English
  4. Bill : Confucian

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

Expansion Rules for Acyclic TBoxes

- One step beyond simple axioms
  - $A \equiv C$ (the same as $A \sqsubseteq C, C \sqsubseteq A$)

\[\text{The } \equiv_1\text{-rule}\]
\[
\text{Condition: } a:A, A \equiv C \in \mathcal{T}, \text{ and } a:C \notin A
\]
\[
\text{Action: } \mathcal{A} \rightarrow \mathcal{A} \cup \{a:C\}
\]

\[\text{The } \equiv_2\text{-rule}\]
\[
\text{Condition: } a:\neg A \in \mathcal{A}, A \equiv C \in \mathcal{T}, \text{ and } a:\neg C \notin \mathcal{A}
\]
\[
\text{Action: } \mathcal{A} \rightarrow \mathcal{A} \cup \{a:\neg C\}
\]

[credit: F Baader]
Expansion Rules for GCI

- **General Class Inclusion (GCI)**
  - $C \sqsubseteq D$, where $C$ is a class description and not a named class
  - Idea: turn the left handside into $T$ (top), thus it is applicable to every individual in the tableau
  - How?
    - Since $C \sqcup \neg C$ is equivalent to $T$, we can turn $C \sqsubseteq D$ into $T \sqsubset D \cup \neg C$

- **GCI is expensive to deal with**
  - since it adds a disjunctive to every individual
- **Don’t forget to use NNF**

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 $O$: $O' = O \cup \{x:C\}$
  - If $O'$ is inconsistent, then the assumption is invalid, so $C$ is **insatisfiable**
  - Otherwise, $C$ is **satisfiable**
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 assertions about x
  - Condition: Sub(y) ⊆ 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

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) ⊆ 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={x}
    - PersonI = {x}
    - friendI = {<x,x>}

Knowledge Graphs
Jeff Z. Pan
Class Subsumption Checking

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

Example: Class Subsumption Checking

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

1. To test the subsumption, we need to check the satisfiability of the class $A \cap \exists r.A \cap \forall r.B \cap \neg (A \cap \exists r.B)$
2. Turn it into NNF: $A \cap \exists r.A \cap \forall r.B \cap \neg (A \cup \forall r. \neg B)$
3. Initial the ABox: $A_1 = \{x: A \cap \exists r.A \cap \forall r.B \cap (\neg A \cup \forall r. \neg B)\}$
4. Apply $\cap$-rule: $A_2 = A_1 \cup \{x:A, x: \exists r.A, x: \forall r.B, x: \neg A \cup \forall r. \neg B\}$
5. Apply $\exists$-rule: $A_3 = A_2 \cup \{x:x1:A\}$
6. Apply $\forall$-rule: $A_4 = A_3 \cup \{x1:B\}$
7. Apply $\cup$-rule: $A_5 = A_4 \cup \{x: \neg A\}$, clash
8. Backtrack, $A_5' = A_4 \cup \{x: \forall r. \neg B\}$
9. Apply $\forall$-rule: $A_6 = A_5' \cup \{x1: \neg B\}$, 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

Example

- If an ontology O entails Male (nicolas)
  - then in every interpretation I of O
  - we have nicolas^I ∈ 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)
Class Instance Checking

- **Question:** given the following ontology $O$,
  - $\text{OldLady} \sqsubseteq \forall \text{hasPet.Car}$
  - $\text{OldLady}(\text{Minnie})$
  - $\text{hasPet} (\text{Minnie}, \text{Tom})$
- Does $O$ entail $\text{Tom}: \text{Cat}$?

1. Add $\text{Tom}: \neg \text{Cat}$ into the ontology $O$
2. Initial the tableau: $A = \{\text{Minnie}: \text{OldLady, } \langle \text{Minnie, Tom} \rangle : \text{hasPet, Tom}: \neg \text{Cat}\}$
3. Apply $S$-rule on axiom 1 with $\text{Minnie}$: $A_1 = A \cup \{\text{Minnie}: \text{hasPet.Car}\}$
4. Apply $\forall$-rule on $\text{Minnie}$: $A_2 = A_1 \cup \{\text{Tom}: \text{Cat}\}$ clash, not backtractable
5. Thus the extended ontology is **inconsistent**
6. And the entailment holds

Example: Consistency Checking

$T := \{\neg (A \cup B) \sqsubseteq \bot, \ A \sqsubseteq \neg B \land \exists r.B, \ D \sqsubseteq \forall r.A, \ B \sqsubseteq \neg A \land \exists r.A\}$

1. Rewrite the first axiom into $T \sqsubseteq A \cup B$
2. Since any individual (such as $x$) is an instance of $T$, $x$ must be an instance of $A \cup B$
3. Initialise the tableau: $A_1 = \{x: A \cup B\}$
4. Apply $L$-rule on $x$: $A_2 = A_1 \cup \{x: A\}$
5. Apply $S$-rule on axiom 2 with $x$: $A_3 = A_2 \cup \{x: \neg B \land \exists r.B\}$
6. Apply $\forall$-rule on $x$: $A_4 = A_3 \cup \{x: \neg B, x: \exists r.B\}$
7. Apply $\exists$-rule on $x$: $A_5 = A_4 \cup \{<x, x_1> : r, x_1: B\}$
8. Apply GCI-rule on axiom 1 with $x_1$: $A_6 = A_5 \cup \{x_1: A \cup B\}$
9. Apply $L$-rule on $x_1$: $A_7 = A_6 \cup \{x_1: B\}$
10. Apply $S$-rule on axiom 4 with $x_1$: $A_8 = A_7 \cup \{\neg A \land \exists r.A\}$
11. Apply $\forall$-rule on $x_1$: $A_9 = A_8 \cup \{\neg A, \exists r.A\}$
12. Apply $\exists$-rule on $x_1$: $A_{10} = A_9 \cup \{<x_1, x_2> : r, x_2: A\}$
13. $x_2$ is blocked by $x$
14. All axioms have been dealt with, and there is no contradiction, so that the TBox is consistent
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:

\[ A_{ct} = \{ a : A \sqcap \exists s.F, \quad (a, b) : s, \]
\[ a : \forall s.(\neg F \sqcup \neg B), \quad (a, c) : r, \]
\[ b : B, \quad c : C \sqcap \exists s.D \}. \]