ulm university universität uulm

Semantic Web Grundlagen

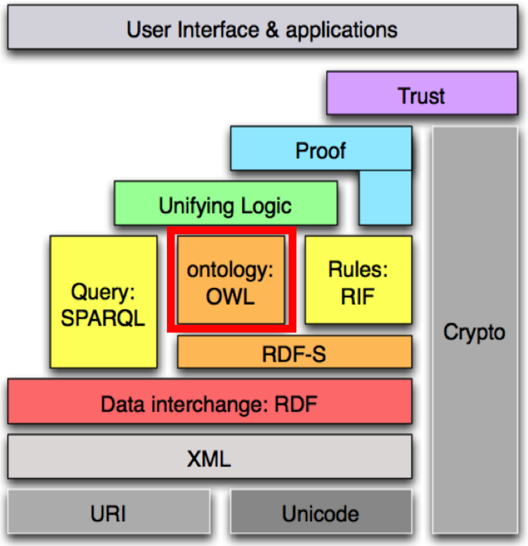
Tableau Prozeduren, Blocking & Unravelling

Birte Glimm
Institut für Künstliche Intelligenz | 05. Dez 2011

Organisatorisches: Inhalt

Einleitung und XML	17. Okt	SPARQL Syntax & Intuition	12. Dez
Einführung in RDF	20. Okt	Übung 4	15. Dez
RDF Schema	24. Okt	SPARQL Semantik	19. Dez
fällt aus	27. Okt	SPARQL 1.1	22. Dez
Logik – Grundlagen	31. Okt	Übung 5	9. Jan
Übung 1	3. Nov	SPARQL Entailment	12. Jan
Semantik von RDF(S)	7. Nov	SPARQL Implemetierung	16. Jan
RDF(S) & Datalog Regeln	10. Nov	Abfragen & RIF	19. Jan
OWL Syntax & Intuition	14. Nov	Übung 6	23. Jan
Übung 2	17. Nov	Ontology Editing	26. Jan
OWL & BLs	21. Nov	Ontology Engineering	30. Jan
OWL 2	24. Nov	Linked Data	2. Feb
Tableau	28. Nov	Übung 7	6. Feb
Übung 3	1. Dez	SemWeb Anwendungen	9. Feb
Blocking & Unravelling	5. Dez	Wiederholung	13. Feb
Hypertableau	8. Dez	Übung 8	16. Feb

OWL 2



Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Tableau Algorithmus für $\mathcal{ALC}$ Konzepte und TBoxen

- Testen der Erfüllbarkeit von C durch Konstruktion einer Abstraktion eines Modells $\mathcal{I}$ so dass $C^{\mathcal{I}} \neq \emptyset$
- TBox wird in ein Konzept $C_{\mathcal{T}}$ internalisiert
- Konzepte in Negationsnormalform (NNF) $\leadsto$ vereinfacht Regeln
- Tableau (Modellabstraktion) entspricht einem Graph/Baum $G = \langle V, E, L \rangle$
- Initialisiere G mit einem Knoten v mit $L(v) = \{C\}$
- Erweitere G durch Anwendung von Tableau Regeln
  - $\sqcup$-Regel ist nichtdeterministisch (wir raten)
- Tableauzweig geschlossen falls G einen atomaren Widerspruch enthält (Clash)
- Tableaukonstruktion erfolgreich, wenn keine Regeln anwendbar sind und kein Widerspruch vorliegt
- C is erfüllbar g.d.w. es eine erfolgreiche Tableaukonstruktion gibt

Tableau Regeln für $\mathcal{ALC}$ Konzepte und TBoxen

$\sqcap$-Regel: Für ein $v \in V$ mit $C \sqcap D \in L(v)$ und $\{C, D\} \not\subseteq L(v)$, setzte $L(v) := L(v) \cup \{C, D\}$.

$\sqcup$-Regel: Für ein $v \in V$ mit $C \sqcup D \in L(v)$ und $\{C, D\} \cap L(v) = \emptyset$, setzte $L(v) := L(v) \cup \{X\}$ für ein $X \in \{C, D\}$.

$\exists$-Regel: Für ein nicht blockiertes $v \in V$ mit $\exists r.C \in L(v)$, so dass es keinen r-Nachfolger v' für v mit $C \in L(v')$ gibt, setzte $V = V \cup \{v'\}$, $E = E \cup \{\langle v, v' \rangle\}$, $L(v') := \{C\}$ und $L(v, v') := \{r\}$ für v' ein neuer Knoten.

$\forall$-Regel: Für ein $v \in V$ mit r-Nachbarn v', $\forall r.C \in L(v)$ und $C \notin L(v')$, setze $L(v') := L(v') \cup \{C\}$.

$\mathcal{T}$-Regel: Für ein $v \in V$ mit $C_{\mathcal{T}} \notin L(v)$, setze $L(v) := L(v) \cup \{C_{\mathcal{T}}\}$.

Blocking/Blockierung

Ein Knoten $v \in V$ blockiert einen Knoten $v' \in V$ direkt, wenn:

1. v' von v erreichbar ist,
2. $L(v') \subseteq L(v)$; und
3. es keinen direkt blockierten Knoten v'' gibt so dass v' von v'' erreichbar ist.

Ein Knoten $v' \in V$ ist blockiert wenn entweder

1. v' direkt blockiert ist oder
2. es einen direkt blockierten Knoten v gibt so dass v' von v erreichbar ist.

Tableau Algorithmus Beispiel

Beispiel: Sei $\mathcal{T} = \{A \sqsubseteq B \sqcap \exists r.C, B \equiv C \sqcup D, C \sqsubseteq \exists r.D\}$. Ist A erfüllbar bzgl. $\mathcal{T}$?

Normalisierung:
$\mathcal{T}' = \{A \sqsubseteq B, A \sqsubseteq \exists r.C, B \sqsubseteq C \sqcup D, C \sqcup D \sqsubseteq B, C \sqsubseteq \exists r.D\}$

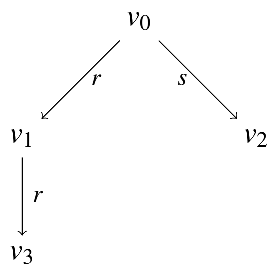
Normalisierung II:
$\mathcal{T}' = \{A \sqsubseteq B, A \sqsubseteq \exists r.C, B \sqsubseteq C \sqcup D, C \sqsubseteq B, D \sqsubseteq B, C \sqsubseteq \exists r.D\}$

$C_{\mathcal{T}} =$
$(\neg A \sqcup B) \sqcap (\neg A \sqcup \exists r.C) \sqcap (\neg B \sqcup C \sqcup D) \sqcap (\neg C \sqcup B) \sqcap (\neg D \sqcup B) \sqcap (\neg C \sqcup \exists r.D)$

Tableau Algorithmus Beispiel

$C_{\mathcal{T}} =$
$(\neg A \sqcup B) \sqcap (\neg A \sqcup \exists r.C) \sqcap (\neg B \sqcup C \sqcup D) \sqcap (\neg C \sqcup B) \sqcap (\neg D \sqcup B) \sqcap (\neg C \sqcup \exists r.D)$

Wir erhalten das folgende widerspruchsfreie Tableau:



$$L(v_0) = \{A, C_{\mathcal{T}}, \ldots, B, \exists r.C, C, \neg D, \exists r.D\}$$
$$L(v_1) = \{C, C_{\mathcal{T}}, \ldots, \neg A, B, \exists r.D\}$$
$$L(v_2) = \{D, C_{\mathcal{T}}, \ldots, \neg A, \neg C, B\}$$
$$L(v_3) = \{D, C_{\mathcal{T}}, \ldots, \neg A, \neg C, B\}$$

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Behandlung von ABoxen

Um auch eine ABox $\mathcal{A}$ zu berücksichtigen, initialisiere G so dass

- V einen Knoten v_a für jedes Individuum a in $\mathcal{A}$ enthält
- $L(v_a) = \{C \mid C(a) \in \mathcal{A}\}$
- $\langle v_a, v_b \rangle \in E$ g.d.w. $r(a,b) \in \mathcal{A}$

Die Regeln können dann auf den initialisierten Graphen G angewendet werden

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Tableau für $\mathcal{ALC}$ mit Inversen Rollen

Um auch eine inverse Rollen zu berücksichtigen, sind folgende Änderungen nötig

1. Markierungen für Kanten können auch inverse Rollen (r^-) enthalten,
2. Ein Knoten v' ist ein r-Nachbar von Knoten v wenn entweder
   - v' ein r-Nachfolger von v ist oder
   - v ein r^--Nachfolger von v' ist
3. ersetze r-Nachfolger in der $\forall$- und der $\exists$-Regel mit r-Nachbar

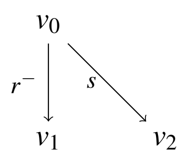
Die $\exists$-Regel erzeugt weiterhin

- einen r-Nachfolger für ein Konzept $\exists r.C$ wenn kein r-Nachbar existiert
- einen r^--Nachfolger für ein Konzept $\exists r^-.C$ wenn kein r^--Nachbar existiert

Tableau Beispiel mit Inversen

Beispiel: Ist A erfüllbar bzgl. $\mathcal{T}$?

$$\mathcal{T} = \{A \equiv \exists r^-.A \sqcap (\forall r.(\neg A \sqcup \exists s.B))\}$$
$$C_\mathcal{T} = (\neg A \sqcup \exists r^-.A) \sqcap (\neg A \sqcup \forall r.(\neg A \sqcup \exists s.B)) \sqcap (\forall r^-.(\neg A) \sqcup \exists r.(A \sqcap \forall s.(\neg B)) \sqcup A)$$



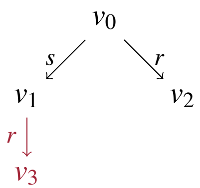
$$L(v_0) = \{A, C_\mathcal{T}, \exists r^-.A, \forall r.(\neg A \sqcup \exists s.B), \neg A \sqcup \exists s.B, \exists s.B\}$$
$$L(v_1) = \{A, C_\mathcal{T}, \exists r^-.A, \forall r.(\neg A \sqcup \exists s.B)\}$$
$$L(v_2) = \{B, C_\mathcal{T}, \neg A, \forall r^-.A\}$$

v_0 blockiert v_1

Ist der Algorithmus daher korrekt? Nein!

Tableau Beispiel mit Inversen II

Beispiel: Ist $C \sqcap \exists s.C$ erfüllbar bzgl. $\mathcal{T}$?



$$\mathcal{T} = \{\top \sqsubseteq \forall r^-.(\forall s^-.(\neg C)) \sqcap \exists r.C\}$$
$$C_\mathcal{T} = \forall r^-.(\forall s^-.(\neg C)) \sqcap \exists r.C$$

$$L(v_0) = \{C, \exists s.C, C_\mathcal{T}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C, \forall s^-.(\neg C)\} \cup \{\neg C\}$$
$$L(v_1) = \{C, C_\mathcal{T}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C\} \cup \{\forall s^-.(\neg C)\}$$
$$L(v_2) = \{C, C_\mathcal{T}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C\}$$

v_0 blockiert v_1 und v_2 aber

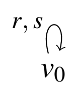
$$L(v_3) = \{C, C_\mathcal{T}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C\}$$

Korrektheit kann wiederhergestellt werden, wenn Teilmengen-Blockierung durch Gleichheits-Blockierung ersetzt wird, d.h., ersetzte $L(v') \subseteq L(v)$ durch $L(v') = L(v)$

Modellkonstruktion für Tableau Beispiel mit Inversen II

Wir können auch nicht wie bisher ein zyklisches Modell bauen!

Beispiel: Ist $C \sqcap \exists s.C$ erfüllbar bzgl. $\mathcal{T}$?

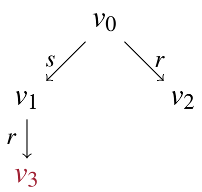


$$\mathcal{T} = \{\top \sqsubseteq \forall r^-.(\forall s^-.(\neg C)) \sqcap \exists r.C\}$$
$$C_\mathcal{T} = \forall r^-.(\forall s^-.(\neg C)) \sqcap \exists r.C$$

$$L(v_0) = \{C, \exists s.C, C_\mathcal{T}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C, \forall s^-.(\neg C)\} \cup \{\neg C\}$$

Tableau Beispiel mit Inversen und Gleichheits-Blockierung

Beispiel: Ist $C \sqcap \exists s.C$ erfüllbar bzgl. $\mathcal{T}$?



$$\mathcal{T} = \{\top \sqsubseteq \forall r^-.(\forall s^-.(\neg C)) \sqcap \exists r.C\}$$
$$C_{\mathcal{T}} = \forall r^-.(\forall s^-.(\neg C)) \sqcap \exists r.C$$

$L(v_0) = \{C, \exists s.C, C_{\mathcal{T}}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C, \forall s^-.(\neg C)\} \cup \{\neg C\}$

$L(v_1) = \{C, C_{\mathcal{T}}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C\} \cup \{\forall s^-.(\neg C)\}$

$L(v_2) = \{C, C_{\mathcal{T}}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C\}$

v_1 blockiert v_2 (Gleiche Markierung)

$L(v_3) = \{C, C_{\mathcal{T}}, \forall r^-.(\forall s^-.(\neg C)), \exists r.C\}$

~~v_1 blockiert v_3 aber $\forall$-Regel anwendbar~~

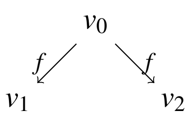
Nun wird die Unerfüllbarkeit erkannt!

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Tableau mit Funktionalen Rollen

Beispiel: Ist A erfüllbar bzgl. $\mathcal{T}$?
Bemerkung: $\top \sqsubseteq \leqslant 1 f$ ist äquivalent zu Func(f)



$$\mathcal{T} = \{A \sqsubseteq \exists f.B \sqcap \exists f.(\neg B), \top \sqsubseteq \leqslant 1 f\}$$
$$C_{\mathcal{T}} = (\neg A \sqcup (\exists f.B \sqcap \exists f.(\neg B))) \sqcap \leqslant 1 f$$

$L(v_0) = \{A, C_{\mathcal{T}}, \ldots, \exists f.B, \exists f.(\neg B), \leqslant 1 f\}$

$L(v_1) = \{B, C_{\mathcal{T}}, \ldots, \neg A, \leqslant 1 f\}$

$L(v_2) = \{\neg B, C_{\mathcal{T}}, \ldots, \neg A, \leqslant 1 f\}$

Funktionalität erfordert dass $v_1 = v_2$!

$\rightsquigarrow$ Für die Behandlung von funktionalen Rollen brauchen wir eine neue Tableau Regel!

Tableau Regeln für $\mathcal{ALCIF}$ Konzepte und TBoxen

$\sqcap$-Regel:	Für ein $v \in V$ mit $C \sqcap D \in L(v)$ und $\{C, D\} \not\subseteq L(v)$, setzte $L(v) := L(v) \cup \{C, D\}$.
$\sqcup$-Regel:	Für ein $v \in V$ mit $C \sqcup D \in L(v)$ und $\{C, D\} \cap L(v) = \emptyset$, setzte $L(v) := L(v) \cup \{X\}$ für ein $X \in \{C, D\}$.
$\exists$-Regel:	Für ein nicht blockiertes $v \in V$ mit $\exists r.C \in L(v)$, so dass es keinen r-Nachfolger v' für v mit $C \in L(v')$ gibt, setzte $V = V \cup \{v'\}$, $E = E \cup \{\langle v, v' \rangle\}$, $L(v') := \{C\}$ und $L(v, v') := \{r\}$ für v' ein neuer Knoten.
$\forall$-Regel:	Für ein $v \in V$ mit r-Nachbarn v', $\forall r.C \in L(v)$ und $C \notin L(v')$, setze $L(v') := L(v') \cup \{C\}$.
$\leqslant 1$-Regel:	Für eine funktionale Rolle f und ein $v \in V$ mit zwei f-Nachbarn v_1 und v_2, merge(v_1, v_2).
$\mathcal{T}$-Regel:	Für ein $v \in V$ mit $C_{\mathcal{T}} \notin L(v)$, setze $L(v) := L(v) \cup \{C_{\mathcal{T}}\}$.

Mergen von Knoten

Wir definieren merge(v_1, v_2) wie folgt:

- wenn v_1 ein Vorfahre (ancestor) von v_2 ist, setze $v_i = v_1$ und $v_o = v_2$;
- sonst, setze $v_i = v_2$ und $v_o = v_1$.

Setze $L(v_i) = L(v_i) \cup L(v_o)$ und prune(v_o).

Wobei prune(v_o) definiert ist als:

- $V_o = \{v \mid v$ gehört zum Teilbaum mit Wurzel $v_o\}$,
- setze $V = V \setminus V_o$ und $E = E \setminus \{\langle v, v_o\rangle \mid v_o \in V_o, \langle v, v_o\rangle \in E\}$.

Tableau mit Funktionalen Rollen

Beispiel: Ist $\exists f.A$ erfüllbar bzgl. $\mathcal{T}$?

$$\begin{tikzcd} v_0 \arrow[d, "f"'] \\ v_1 \arrow[d, "f"'] \\ v_2 \end{tikzcd}$$

$$\mathcal{T} = \{A \sqsubseteq \exists f.A, \top \sqsubseteq \leqslant 1 f^-\}$$
$$C_{\mathcal{T}} = (\neg A \sqcup \exists f.A) \sqcap \leqslant 1 f^-$$

$$L(v_0) = \{\exists f.A, C_{\mathcal{T}}, \neg A, \leqslant 1 f^-\}$$
$$L(v_1) = \{A, C_{\mathcal{T}}, \exists f.A, \leqslant 1 f^-\}$$
$$L(v_2) = \{A, C_{\mathcal{T}}, \exists f.A, \leqslant 1 f^-\}$$

v_1 blockiert v_2, aber zyklische Modellkonstruktion funktioniert nicht (Funktionalität verletzt)!

$$\begin{tikzcd} v_0 \arrow[d, "f"'] \\ v_1 \arrow[loop below, "f"] \end{tikzcd}$$

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Unravelling

Ziel: Wir bauen ein unendliches Modell
Wie? Jeder blockierte Knoten wird durch den Teilbaum ersetzt, der als Wurzel den Blockierer hat.

$$\begin{tikzcd} v_0 \arrow[d, "f"'] \\ v_1 \arrow[d, "f"'] \\ v_2 \arrow[d, "f"'] \\ v_1'' \arrow[d, "f"'] \\ v_1''' \arrow[d, dotted, dash, "f"'] \\ {} \end{tikzcd}$$

$$L(v_0) = \{\exists f.A, C_{\mathcal{T}}, \neg A, \leqslant 1 f^-\}$$
$$L(v_1) = \{A, C_{\mathcal{T}}, \exists f.A, \leqslant 1 f^-\}$$
$$L(v_2) = \{A, C_{\mathcal{T}}, \exists f.A, \leqslant 1 f^-\}$$

v_1 blockiert v_2

Blocking: Inverse und Funktionale Rollen

Beispiel: Ist $\neg C \sqcap \exists f^-.D$ erfüllbar bzgl. $\mathcal{T}$?

$$\mathcal{T} = \{D \sqsubseteq C \sqcap \exists f.(\neg C) \sqcap \exists f^-.D, \top \sqsubseteq \leqslant 1 f\}$$
$$C_{\mathcal{T}} = (\neg D \sqcup (C \sqcap \exists f.(\neg C) \sqcap \exists f^-.D)) \sqcap \leqslant 1 f$$

$$\begin{array}{c} v_0 \\ f^- \downarrow \\ v_1 \\ f^- \downarrow \\ v_2 \\ f^- \downarrow \\ v_1'' \end{array}$$

$$L(v_0) = \{\neg C, \exists f^-.D, C_{\mathcal{T}}, \ldots, \neg D, \leqslant 1 f\}$$
$$L(v_1) = \{D, C_{\mathcal{T}}, \ldots, C, \exists f.(\neg C), \exists f^-.D, \leqslant 1 f\}$$
$$L(v_2) = \{D, C_{\mathcal{T}}, \ldots, C, \exists f.(\neg C), \exists f^-.D, \leqslant 1 f\}$$

v_1 blockiert v_2(gleiche Markierung) aber

$$L(v_1'') = \{D, C_{\mathcal{T}}, \ldots, C, \exists f.(\neg C), \exists f^-.D, \leqslant 1 f\}$$

Aber wir können kein Modell mehr bauen (weder zyklisch noch unendlich)!

Pairwise Blocking/Paarweise Blockierung

Ein Knoten x mit Vorgänger x' blockiert einen Knoten y mit Vorgänger y' direkt, wenn:

1. y von x erreichbar ist,
2. $L(x) = L(y)$, $L(x') = L(y')$ und $L(x', x) = L(y', y)$; und
3. es gibt keinen direkt blockierten Knoten z so dass y von z erreichbar ist.

Ein Knoten $y \in V$ ist blockiert wenn entweder

1. y direkt blockiert ist oder
2. es einen direkt blockierten Knoten x gibt, so dass y von x erreichbar ist.

Paarweise Blockierung: Inverse und Funktionale Rollen

Beispiel: Ist $\neg C \sqcap \exists f^-.D$ erfüllbar bzgl. $\mathcal{T}$?

$$\mathcal{T} = \{D \sqsubseteq C \sqcap \exists f.(\neg C) \sqcap \exists f^-.D, \top \sqsubseteq \leqslant 1 f\}$$
$$C_{\mathcal{T}} = (\neg D \sqcup (C \sqcap \exists f.(\neg C) \sqcap \exists f^-.D)) \sqcap \leqslant 1 f$$

$$\begin{array}{c} v_0 \\ f^- \downarrow \\ v_1 \\ f^- \downarrow \\ v_2 \\ f \downarrow \\ v_3 \end{array}$$

$$L(v_0) = \{\neg C, \exists f^-.D, C_{\mathcal{T}}, \ldots, \neg D, \leqslant 1 f\}$$
$$L(v_1) = \{D, C_{\mathcal{T}}, \ldots, C, \exists f.(\neg C), \exists f^-.D, \leqslant 1 f\}$$
$$L(v_2) = \{D, C_{\mathcal{T}}, \ldots, C, \exists f.(\neg C), \exists f^-.D, \leqslant 1 f\}$$

v_1 kann v_2 nicht paarweise blockieren

$$L(v_3) = \{\neg C, C_{\mathcal{T}}, \ldots, \neg D, \leqslant 1 f\}$$

v_3 wird in v_1 zusammengeschlossen

$$L(v_1) = L(v_1) \cup L(v_3) \supseteq \{\neg D, D\}$$

Nun wird der Widerspruch erkannt!

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Optimierungen

- Naive Implementierung nicht performant genug
  - $\mathcal{T}$-Regel fügt eine Disjuktion pro Axiom zur Markierung jedes Knotens
  - Ontologien enthalten oft > 1.000 Axiome und Tableau können tausende Knoten enthalten
- Realistische Implementierungen wenden viele Optimierungen an
  - (Lazy) unfolding
  - Absorbtion
  - Dependency directed backtracking
  - Vereinfachung und Normalisierung
  - Caching
  - Heuristiken
  - ...

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Unfolding

- $\mathcal{T}$-Regel ist unnötig wenn $\mathcal{T}$ unfoldable ist, d.h., jedes Axiom ist:
  - Definitiorisch: Form $A \sqsubseteq C$ oder $A \equiv C$ für A ein Konzeptname
    ($A \equiv C$ entspricht $A \sqsubseteq C$ und $C \sqsubseteq A$)
  - Azyklisch: C verweist weder direkt noch indirekt auf A
  - Eindeutig: nur ein solches Axiom existiert für jeden Konzeptnamen A
- Wenn $\mathcal{T}$ unfoldable ist, kann man die TBox in ein Konzept “einfalten” (unfold)

Unfolding Beispiel

- Wir testen Erfüllbarkeit von A bzgl. TBox $\mathcal{T}$

	$\mathcal{T}$:
A	$A \sqsubseteq B \sqcap \exists r.C$
$\leadsto A \sqcap B \sqcap \exists r.C$	$B \equiv C \sqcup D$
$\leadsto A \sqcap (C \sqcup D) \sqcap \exists r.C$	$C \sqsubseteq \exists r.D$
$\leadsto A \sqcap ((C \sqcap \exists r.D) \sqcup D) \sqcap \exists r.(C \sqcap \exists r.D)$	

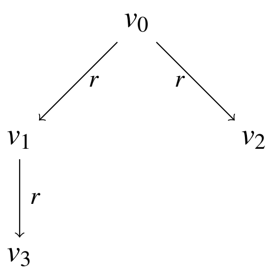
- A ist erfüllbar bzgl. $\mathcal{T}$ g.d.w.

$$A \sqcap ((C \sqcap \exists r.D) \sqcup D) \sqcap \exists r.(C \sqcap \exists r.D)$$

erfüllbar bzgl. der leeren TBox ist

Tableau Algorithmus Beispiel mit Unfolding

Wir erhalten das folgende widerspruchsfreie Tableau für die Erfüllbarkeit von $U = A \sqcap ((C \sqcap \exists r.D) \sqcup D) \sqcap \exists r.(C \sqcap \exists r.D)$:



$$L(v_0) = \{U, A, (C \sqcap \exists r.D) \sqcup D, \exists r.(C \sqcap \exists r.D), C \sqcap \exists r.D, C, \exists r.D\}$$
$$L(v_1) = \{C \sqcap \exists r.D, C, \exists r.D\}$$
$$L(v_2) = \{D\}$$
$$L(v_3) = \{D\}$$

Nur noch eine disjunktive Entscheidung!

Lazy Unfolding

- Bilden der NNF zusammen mit Unfolding kann die Effizienz beeinträchtigen, z.B.:
  - Erfüllbarkeit von $C \sqcap \neg C$ bzgl. $\mathcal{T} = \{C \sqsubseteq A \sqcap B\}$
  - Unfolding: $C \sqcap A \sqcap B \sqcap \neg(C \sqcap A \sqcap B)$
  - NNF + Unfolding: $C \sqcap A \sqcap B \sqcap (\neg C \sqcup \neg A \sqcup \neg B)$
- Besser NNF und Unfolding bei Bedarf anwenden über entsprechende Regeln:
  - $A \equiv C \rightsquigarrow A \sqsubseteq C$ und $A \sqsupseteq C$

$\sqsubseteq$-Regel: Für $v \in V$ so dass $A \sqsubseteq C \in \mathcal{T}$, $A \in L(v)$ und $C \notin L(v)$ setze $L(v) := L(v) \cup C$.

$\sqsupseteq$-Regel: Für $v \in V$ so dass $A \sqsupseteq C \in \mathcal{T}$, $\neg A \in L(v)$ und $\neg C \notin L(v)$ setze $L(v) := L(v) \cup \{\neg C\}$.

$\neg$-Regel: Für $v \in V$ so dass $\neg C \in L(v)$ und NNF$(\neg C) \notin L(v)$, setze $L(v) := L(v) \cup \{\text{NNF}(\neg C)\}$.

Agenda

Absorbierung

- Was, wenn $\mathcal{T}$ nicht unfoldable ist?
  - Teile $\mathcal{T}$ in $\mathcal{T}_u$ (unfoldable Teil) und $\mathcal{T}_g$ (GCIs, nicht unfoldable)
  - $\mathcal{T}_u$ wird mittels $\sqsubseteq$- und $\sqsupseteq$-Regeln behandelt
  - $\mathcal{T}_g$ wird mittels $\mathcal{T}$-Regel behandelt
- Absorbierung verkleinert $\mathcal{T}_g$ und vergrößert $\mathcal{T}_u$
  1. Nehme ein Axiom aus $\mathcal{T}_g$, z.B., $A \sqcap B \sqsubseteq C$
  2. Transformiere das Axiom: $A \sqsubseteq C \sqcup \neg B$
  3. Wenn $\mathcal{T}_u$ ein Axiom der Form $A \equiv D$ $\quad(A \sqsubseteq D$ and $D \sqsupseteq A)$ enthält, dann ist $A \sqsubseteq C \sqcup \neg B$ nicht absorbierbar; $A \sqsubseteq C \sqcup \neg B$ verbleibt in $\mathcal{T}_g$
  4. sonst, falls $\mathcal{T}_u$ ein Axiom der Form $A \sqsubseteq D$ enthält, dann absorbiere $A \sqsubseteq C \sqcup \neg B$ mit Ergebnis $A \sqsubseteq D \sqcap (C \sqcup \neg B)$
  5. sonst, verschiebe $A \sqsubseteq C \sqcup \neg B$ in $\mathcal{T}_u$
- Wenn $A \equiv D \in \mathcal{T}_u$, versuche das Umschreiben/Absorbieren mit anderen Axiomen in $\mathcal{T}_u$
- Nichtdeterministisch: $B \sqsubseteq C \sqcup \neg A$ auch möglich

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Dependency Directed Backtracking

- Trotz diese Optimierungen, Suchraum oft zu groß
- Sei $v \in V$ mit
  $(C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n) \sqcap \exists r.\neg A \sqcap \forall r.A \in L(v)$

$v \xrightarrow{r} w$

$\sqcap$-Regel	L(v)	:=	$L(v) \cup \{(C_1 \sqcup D_1), \ldots, (C_n \sqcup D_n), \exists r.\neg A, \forall r.(A \sqcap B)\}$
$\sqcup$-Regel	L(v)	:=	$L(v) \cup \{C_1\}$
$\vdots$	$\vdots$		$\vdots$
~~$\sqcup$-Regel~~	~~L(v)~~	:=	$\cancel{L(v) \cup \{C_n\}}$
~~$\exists$-Regel~~	~~L(w)~~	:=	$\cancel{\{\neg A\}}$
~~$\forall$-Regel~~	~~L(w)~~	:=	$\cancel{\{\neg A, A\}}$ ~~*clash*~~
$\sqcup$-Regel	L(v)	:=	$L(v) \cup \{D_n\}$
$\exists$-Regel	L(w)	:=	$\{\neg A\}$
$\forall$-Regel	L(w)	:=	$\{\neg A, A\}$ *clash*

- Exponentiel großer Suchraum wird durchsucht

Dependency Directed Backtracking

- Ziel ist es, schlechte Verzweigungsentscheidungen schnell zu erkennen und nicht zu wiederholen
- Häufigste Technik: backjumping
- Backjumping funktioniert grob wie folgt:
  - Konzepte in der Knotenmarkierung werden mit einer Menge von Integers markiert (dependency set) über das die Herkunft des Konzeptes ersichtlich ist
  - Initial sind Konzepte mit $\emptyset$ markiert
  - Tableau Regeln kombinieren und erweitern diese Markierungen
  - $\sqcup$-Regel fügt $\{d\}$ zur Markierung mit d die $\sqcup$-Tiefe (Anzahl der $\sqcup$-Regel Anwendungen bisher)
  - Markierungen geben bei einem Widerspruch Auskunft über die Herkunft der widersprüchlichen Konzepte
  - Springe direkt zur letzten relevanten $\sqcup$-Regel Anwendung
- Irrelevanter Teil des Suchraums wird nicht betrachtet

Dependency Directed Backtracking Beispiel

$(C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n) \sqcap \exists r.\neg A \sqcap \forall r.A \in L(v)$ markiert mit $\emptyset$

$v \xrightarrow{r} w$

$\sqcap$-Regel	L(v)	:=	$L(v) \cup \{(C_1 \sqcup D_1), \ldots, (C_n \sqcup D_n), \exists r.\neg A, \forall r.(A \sqcap B)\}$	alle mit $\emptyset$
$\sqcup$-Regel	L(v)	:=	$L(v) \cup \{C_1\}$	C_1 markiert mit $\{1\}$
$\vdots$	$\vdots$		$\vdots$	
$\sqcup$-Regel	L(v)	:=	$L(v) \cup \{C_n\}$	C_n markiert mit $\{n\}$
$\exists$-Regel	L(w)	:=	$\{\neg A\}$	A, r markiert mit $\emptyset$
$\forall$-Regel	L(w)	:=	$\{\neg A, A\}$ clash	$\neg A$ markiert mit $\emptyset$

- tag$(A) \cup$ tag$(\neg A) = \emptyset$
- Keine der $\sqcup$-Regeln hat etwas zum Widerspruch beigetragen
- Rückgabe falsch (unerfüllbar)

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Andere Optimierungen

- Vereinfachung und Normalisierung
  - Schnelles Erkennen trivialer Widersprüche
  - Normalisierung, z.B., $A \sqcap (B \sqcap C) \equiv \sqcap\{A, B, C\}$, $\forall r.C \equiv \neg\exists r.\neg C$
  - Vereinfachung, z.B., $\sqcap\{A, \ldots, \neg A, \ldots\} \equiv \bot$, $\exists r.\bot \equiv \bot$, $\forall r.\top \equiv \top$
- Caching
  - Verhindert die erneute Konstruktion von gleichen Teilbäumen
  - $L(v)$ initialisiert mit $\{C_1, \ldots, C_n\}$ durch $\exists$- und $\forall$-Regeln
  - Prüfe ob Erfüllbarkeitsstatus gecached ist, sonst
  - Teste Erfüllbarkeit von $C_1 \sqcap \ldots \sqcap C_n$, aktualisiere den Cache
- Heuristiken
  - Versuche gute Ordnung für den “don’t care” Nichtdeterminismus zu finden
  - Z.B., $\sqcap$, $\forall$, $\sqcup$, $\exists$ Ordnung
- ...

Agenda

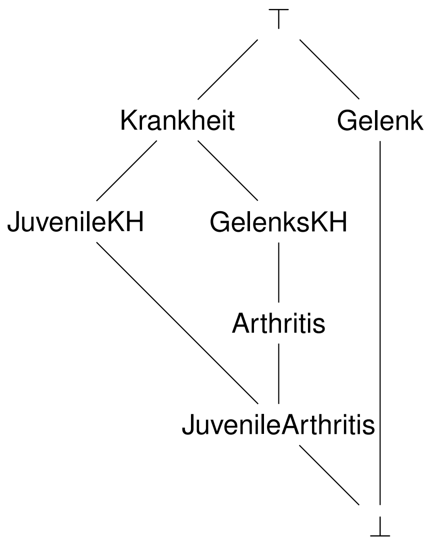
- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Klassifikation Optimieren

Einer der verbreitetsten Aufgaben für das automatische Schlussfolgern ist die Klassifikation

- Berechne alle Unterklassenbeziehungen zwischen Konzeptnamen in $\mathcal{T}$
- Test von $C \sqsubseteq D$ entspricht Erfüllbarkeitstest von der TBox plus ABox $(C \sqcap \neg D)(a)$ also $C(a), (\neg D)(a)$
  - $\rightsquigarrow$ Wenn $\mathcal{T}$ erfüllbar ist: Subsumption hält nicht (wir haben ein Gegenmodell konstruiert)
  - $\rightsquigarrow$ Wenn $\mathcal{T}$ unerfüllbar ist: Subsumption hält (ein Gegenmodell existiert nicht)
- Naiver Ansatz braucht n^2 Vergleiche für n Konzeptnamen
- Üblicherweise gecached im Konzept-Hierarchy Graphen

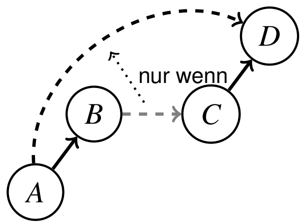
Graph einer Konzept-Hierarchie



Optimierung der Klassifikation

Verbreitetste Technik nennt sich Enhanced Traversal

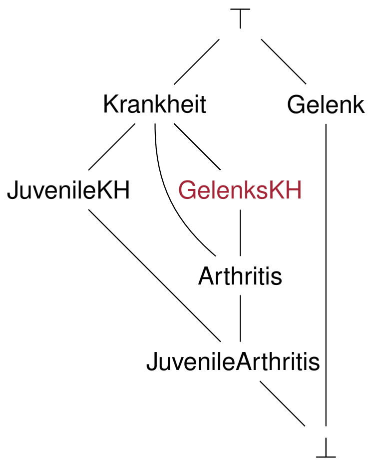
- Die Hierarchie wird inkrementell erstellt durch Einfügen eines Konzepts nach dem anderen
- Top-down Phase: Erkennen von direkten Oberkonzepten
- Bottom-up Phase: Erkennen von direkten Unterkonzepten
- Transitivität von $\sqsubseteq$ wird genutzt um Tests zu vermeiden



- Wenn $A \sqsubseteq B$ gilt und $C \sqsubseteq D$,
- dann $B \sqsubseteq C \longrightarrow A \sqsubseteq D$
- und $A \not\sqsubseteq D \longrightarrow B \not\sqsubseteq C$

Enhanced Traversal Beispiel

Bereits erstellte Hierarchie:



Ziel: Einfügen von JuvenileArthritis

Top-Down Phase:

- GelenksKH $\sqsubseteq^?$ Krankheit
- GelenksKH $\sqsubseteq^?$ JuvenileKH
- GelenksKH $\sqsubseteq^?$ Arthritis
- GelenksKH $\sqsubseteq^?$ Gelenk

Bottom-Up Phase:

- JuvenileArthritis $\sqsubseteq^?$ GelenksKH
- JuvenileKH $\sqsubseteq^?$ GelenksKH
- Arthritis $\sqsubseteq^?$ GelenksKH

Agenda

- Wiederholung Tableaukalkül
- Tableau für $\mathcal{ALC}$ Wissensbasen
- Erweiterung um Inverse Rollen
- Erweiterung um Funktionale Rollen
- Modellkonstruktion mit Unravelling
- Optimierungen
  - Unfolding
  - Absorbierung
  - Dependency Directed Backtracking
  - Weitere Optimierungen
- Klassifikation
- Zusammenfassung

Zusammenfassung

- Wir haben nun einen Tableau Algorithmus für $\mathcal{ALCIF}$ Wissensbasen
  - ABox behandelt wie für $\mathcal{ALC}$
  - Zahlenrestriktionen ähnlich wie Funktionalität bzw. Existenzquantoren
- Terminierung durch Zyklenerkennung
  - Schwieriger je ausdrucksstärker die Logik wird
- Naives Tableauverfahren nicht performant genug
- Diverse Optimierungen verbessern den average case
- Spezielle Verfahren zur Klassifikation
  - Enhanced Traversal
- Tableau bzw. Variationen davon Grundlage der OWL Reasoner