Ausarbeitung zum Seminarvortrag "Tableau-Beweiser für DLs"

Christian Sattler

6. Januar 2010

Beispiele zum gleichzeitige Erläuterung des Tableau-Algorithmus

Vatermörder Wir wollen das Tableau-Verfahren an einigen Beispielen erklärend und gleichzeitig näher erläutern. Betrachten wir noch einmal das Vatermörder-Beispiel aus den Folien mit zugeordneter ABox $\mathcal{A}_{oe}$

hasChild(J, O)	hasChild(J, P)
hasChild(O, P)	hasChild(P, T)
Patricide(O)	$(\neg\text{Patricide})(T)$,

die folgende Situation modellieren soll: **J**ocaste gebährt ihrem Mann Laios, dem König von Theben, ihr gemeinsames Kind **O**edipus (hasChild(J, O)). Dieser tötet später in einem Konflikt, in Unkenntnis seiner wahren Eltern, seinen eigenen Vater, wird also zum Vatermörder (Patricide(O)). Als Lohn für das Besiegen der Sphinx ehelicht er die Witwe **J**ocaste und hat mit ihr den Sohn **P**olyneikes (hasChild(J, P), hasChild(O, P)). Dieser hat in seinem weiteren Leben noch den Sohn **T**hersandros (hasChild(P, T)), später ebenfalls König von Theben und kein Vatermörder ($(\neg\text{Patricide})(T)$).

Betrachten wir das Instance-Checking-Problem

$$\mathcal{A}_{oe} \models \text{Query}(J), \text{ wobei Query} := \exists\text{hasChild}.(\text{Patricide} \sqcap \exists\text{hasChild}.\neg\text{Patricide}),$$

in Worten: Ist es in obiger Situation zwingend der Fall, dass Jocaste das Konzept erfüllt, ein Kind zu haben (hasChild ...), das ein Vatermörder ist (Patricide) und ($\sqcap$) selber ein Kind hat, das kein Vatermörder ist (hasChild.$\neg$Patricide)?

Das Tableau-Verfahren gibt uns zunächst nur einen Algorithmus um die Inkonsistenz bzw. Erfüllbarkeit einer ABox festzustellen. Nun ist Query(J) in der ABox $\mathcal{A}_{oe}$ genau dann zwangsläufig erfüllt, wenn $(\neg\text{Query})(J)$ nie erfüllt ist, also die erweiterte ABox $\mathcal{A}_{query} := \mathcal{A}_{oe} \cup \{(\neg\text{Query})(A)\}$ inkonsistent ist.

Der nächste Schritt besteht darin, sämtliche Konzepte in der ABox in Negationsnormalform zu überführen, d. h. alle Negationen vor primitive Konzepte zu verschieben, denn der Tableau-Algorithmus in unserer Form hat keine Transformationsregel zum Umgang mit allgemeinen Negationen (tatsächlich gibt es auch Varianten von Tableau-Algorithmen, die mit Negationen, und allgemein sogar Implikationen, arbeiten, dafür aber z. B. keine Regeln für $\sqcup$ und $\exists$ haben, also Formeln in einer anderen Art von Normalform betrachten). Glücklicherweise befinden sich bis auf $(\neg\text{Query})(J)$ alle Konzepte in $\mathcal{A}_{query}$ bereits in Negationsnormalform. Wir verschieben nun die Negation in $\neg$Query von außen nach innen, wobei wir beachten, dass dabei jedes logische Konnektiva

durch sein duales Gegenstück ausgetauscht wird und wir die Stabilität $\neg\neg C \to C$ benutzen dürfen:

$$\begin{aligned}
& \neg\exists \text{hasChild}.(\text{Patricide} \sqcap \exists \text{hasChild}.\neg\text{Patricide}) \\
\longrightarrow \quad & \forall \text{hasChild}.\neg(\text{Patricide} \sqcap \exists \text{hasChild}.\neg\text{Patricide}) \\
\longrightarrow \quad & \forall \text{hasChild}.(\neg\text{Patricide} \sqcup \neg\exists \text{hasChild}.\neg\text{Patricide}) \\
\longrightarrow \quad & \forall \text{hasChild}.(\neg\text{Patricide} \sqcup \forall \text{hasChild}.\neg\neg\text{Patricide}) \\
\longrightarrow \quad & \underbrace{\forall \text{hasChild}.(\neg\text{Patricide} \sqcup \forall \text{hasChild}.\text{Patricide})}_{\text{NegQuery}}.
\end{aligned}$$

Wir erhalten eine neue, normalisierte ABox $\mathcal{A}'_{Query} := \mathcal{A}_{oe} \cup \{\text{NegQuery}(J)\}$.

Nun beginnt die eigentliche Phase des Tableau-Verfahrens. Sukzessive werden wir solange bedingte Transformationsregeln der Form $\mathcal{A} \to \mathcal{A}_1, \ldots, \mathcal{A}_k$ auf unsere ABox anwenden, wobei wir die rechte Seite als Menge von Alternativen verstehen, bis wir entweder auf allen Pfaden auf eine *Kollision* (d. h. $C(x)$ und $(\neg C)(x)$, oder $\bot$) stoßen oder keine Regel mehr anwenden können. Die Regeln sind, wie man direkt sehen kann, derart gestaltet, dass für jede Interpretation $\mathcal{I}$ gilt: Genau dann ist $\mathcal{I} \models \mathcal{A}$ (d. h. $\mathcal{I}$ ein Model von $\mathcal{A}$), wenn $\mathcal{I} \models \mathcal{A}_i$ für ein $i \in \{1, \ldots, k\}$. Im ersten Fall ist damit die ursprüngliche ABox inkonsistent, im zweiten Fall haben wir die Formeln in der ABox komplett in ihre Bestandteile zerlegt ohne einen Widerspruch zu finden und können nun an den notwendig gültigen Bestandteilen eine passende Interpretation der Konzepte ablesen (wie wir später sehen werden). Aufgrund obiger semantischer Erhaltungseigenschaft der Transformationsregeln ist dies dann auch eine passende Interpretation für die ursprüngliche ABox.

- Die $\to_\sqcap$-Regel
  Bedingung: $\mathcal{A}$ enthält $(C_1 \sqcap C_2)(x)$, aber nicht sowohl $C_1(x)$ als auch $C_2(c)$
  Aktion: $\mathcal{A} \to \mathcal{A} \cup \{C_1(x), C_2(x)\}$

- Die $\to_\sqcup$-Regel
  Bedingung: $\mathcal{A}$ enthält $(C_1 \sqcap C_2)(x)$, aber nicht $C_1(x)$ oder $C_2(c)$
  Aktion: $\mathcal{A} \to \mathcal{A} \cup \{C_1(x)\}, \mathcal{A} \cup \{C_2(x)\}$

- Die $\to_\forall$-Regel
  Bedingung: $\mathcal{A}$ enthält $(\forall R.C)(x)$ und $R(x, y)$, aber nicht $C(y)$
  Aktion: $\mathcal{A} \to \mathcal{A} \cup \{C(y)\}$

- Die $\to_\exists$-Regel
  Bedingung: $\mathcal{A}$ enthält $(\exists R.C)(c)$ und es gibt noch keinen Individuennamen z, so dass $C(z)$ und $R(x, z)$ in $\mathcal{A}$ liegen
  Aktion: $\mathcal{A} \to \mathcal{A}\{C(y), R(x, y)\}$, wobei y ein neuer Individuenname ist

Die Regeln beziehen sich, wie man sieht, jeweils auf das äußerste syntaktische Konstrukt einer Konzeptinstanz innerhalb der ABox. Nur die $\sqcup$-Regel ist nichtdeterministisch, sie ergibt zwei mögliche ABoxen zum Fortfahren, die wir beide auf Widersprüchlichkeit überprüfen müssen, wenn wir auf die Widersprüchlichkeit der Ausgangs-ABox schließen wollen.

Graphisch lässt sich das Verfahren auf folgende Weise darstellen: Wir starten mit einer ausgewählten Wurzel und notieren die Formeln der ursprünglichen ABox an ihr. Für jede Regelanwendung (die wir entsprechend markieren) zeichnen wir für jede Alternative der Regel einen neuen Knoten über den alten (für deterministische Regeln also nur einen), fügen die jeweils neu gewonne(n) Formel(n) hinzu und arbeiten mit den neuen Knoten weiter. Für einen Knoten ist die zugehörige ABox,

also die Formeln, auf die wir unsere Regeln anwenden dürfen, gegeben durch die Menge der auf dem Pfad zur Wurzel anzutreffenden Formeln.[1] Diese Art der Darstellung erklärt die Namensgebung des Algorithmus, der auch als *analytisches* Tableau bezeichnet wird, da er folgende wichtige *Unterformel-Eigenschaft* aufweist: Der Konzeptteil C jeder neu hinzukommenden Formel $C(x)$ ist syntaktisch Teil des Konzeptteil einer bereits enthaltenen Formel:

$$\dfrac{\dfrac{\dfrac{\bot}{(\neg\text{Patricide})(O)}\quad \text{Patricide}(O)\qquad \dfrac{\dfrac{\dfrac{\bot}{(\neg\text{Patricide})(P)}\quad \text{Patricide}(P)\quad \dfrac{\dfrac{\bot}{\text{Patricide}(T)}\;\; (\neg\text{Patricide})(T)}{(\forall\text{hasChild.Patricide})(P)}\,\to_{\forall}}{\text{Patricide}(P)}\,\to_{\sqcup}}{(\forall\text{hasChild.Patricide})(O)}\,\to_{\forall}}{\dfrac{(\neg\text{Patricide}\sqcup\forall\text{hasChild.Patricide})(P)}{(\neg\text{Patricide}\sqcup\forall\text{hasChild.Patricide})(O)}\,\to_{\forall}}\,\to_{\sqcup}}{\mathcal{A}'_{query} = \left\{\begin{array}{c} \text{hasChild}(J,O), \text{hasChild}(J,P), \text{hasChild}(O,P), \\ \text{hasChild}(P,T), \text{Patricide}(O), (\neg\text{Patricide})(T), \\ (\forall\text{hasChild}.(\neg\text{Patricide}\sqcup\forall\text{hasChild.Patricide}))(J) \end{array}\right\}}\,\to_{\forall}$$

Auf allen Pfaden ergibt sich ein Widerspruch, die ABox $\mathcal{A}'_{query}$, und somit ebenfalls $\mathcal{A}_{query}$, ist also inkonsistent. Damit ist die ursprüngliche Instance-Checking-Fragestellung $\mathcal{A}_{oe} \models \text{Query}(J)$ positiv zu beantworten. Man beachte, dass wir bei den Anwendungen der Regeln freie Wahl haben (Nichtdeterminismus auf dem (Meta-)Level des Algorithmus). Für Korrektheit und Vollständigkeit ist die Reihenfolge jedoch unerheblich, wie uns die semantische Erhaltungseigenschaft der Regeln garantiert. [2] Lediglich die Eigenschaft der Terminierung, die wir für einen totalen Algorithmus benötigen, ist durch diese Variationsmöglichkeit gefährdet (dazu später).

Modellkonstruktion Hier noch ein Beispiel, an dem zu sehen ist, wie der Tableau-Algorithmus als Proof-Witness für die Erfüllbarkeit ein Modell aufbaut: Wir interessieren uns für mögliche Subsumptionsrelationen zwischen den Konzepten C und D mit $C \equiv \forall R.(A \sqcup B)$ und $D \equiv (\forall R.A) \sqcup (\forall R.B)$, d. h. ob sie äquivalent sind ($C \equiv D$) oder eines von ihnen allgemeiner als das andere ist ($C \sqsubseteq D$ bzw. $D \sqsubseteq C$). Äquivalenz bedeutet nichts anderes als die Gültigkeit beider Subsumptionsrelationen, weswegen wir uns auf die Untersuchung dieser beschränken können. Nun bedeutet die Gültigkeit von $C \sqsubseteq D$ nichts anderes als die Unerfüllbarkeit von $C \sqcap \neg D$, in Negationsnormalform $Q \equiv (\forall R.(A \sqcup B)) \sqcap ((\exists R.\neg A) \sqcap (\exists R.\neg B))$. Wir wenden den Tableau-Algorithmus auf die ABox

[1] Alternativ können wir auch an jedem Knoten alle Formeln der zugehörigen ABox notieren. Dies resultiert jedoch in erheblichem zusätzlichen Darstellungsaufwand und einer gewissen Unübersichtlichkeit.

[2] Hier sieht noch auch leicht, dass das durch die Regelanwendungen gegebene Verfahren konfluent (für zwei verschiedene Fortführungen derselben ABox können wir weitere Regelanwendungen vornehmen, durch die wir auf beiden Pfaden zu denselben Mengen an möglichen ABoxen gelangen) ist

$\{Q(x)\}$ mit einem generischen Individuennamen x an:

$$
\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\bot}{A(y)} \quad (\neg A)(y) \qquad \cfrac{\cfrac{\cfrac{A(z) \quad \cfrac{\bot}{B(z)}\ (\neg B)(z)}{(A \sqcup B)(z)}\ \to_\sqcup}{R(x,z), (\neg B)(y)}\ \to_\forall}{B(y)}\ \to_\exists}{(A \sqcup B)(y)}\ \to_\sqcup}{R(x,y), (\neg A)(y)}\ \to_\forall}{(\exists R.\neg A)(x), (\exists R.\neg B)(x)}\ \to_\exists}{(\forall R.(A \sqcup B))(x), ((\exists R.\neg A) \sqcap (\exists R.\neg B))(x)}\ \to_\sqcap}{((\forall R.(A \sqcup B)) \sqcap ((\exists R.\neg A) \sqcap (\exists R.\neg B)))(x)}\ \to_\sqcap
$$

Im mittleren, mit $A(z)$ endenden Pfad können wir keine der Regeln mehr anwenden. Wir versuchen, an den deduzierten primitiven Konzeptrelationen eine Interpretation I als Gegenbeispiel für $C \sqsubseteq D$ abzulesen. Für jedes primitive Konzept C und Individuennamen t wählen wir $t \in C^{\mathcal{I}}$ genau dann, wenn $C(t)$ auf dem Pfad vorkommt, analog für die Rollenrelation. Wir erhalten die Individuenmenge $\Delta^{\mathcal{I}} = \{x, y, z\}$ sowie die Konzept- und Rolleninterpetationen $A^{\mathcal{I}} = \{z\}, B^{\mathcal{I}} = \{y\}$ und $R^{\mathcal{I}} = \{(x, y), (x, z)\}$. Die Tatsache, dass für jedes logische Konnektiva, also für jede nicht-primitive Konzeptformel strukturell eine Transformationsregel in Frage kommt, die zugehörige Bedingung jedoch nie erfüllt ist, ermöglichst es uns nun, induktiv über die Struktur der auf dem Pfad des Tableaus vorkommenden Konzepte zu zeigen, dass unsere Interpretation ein Modell für jede auf dem Pfad vorkommende Formel, also auch unsere ursprüngliche Formel $Q(x)$ ist. Wir zeigen dies ganz allgemein, unabhängig von unserem konkreten Tableau:

Betrachten wir eine Formel $E(t)$ auf dem Pfad und nehmen wir an, dass für alle echten strukturelle Unterkonzepte F von E die Erfüllbarkeit aller auf dem Pfad vorkommenden Instanzen $F(u)$ bereits gezeigt wurde.

Sei C ein primitiver Konzeptname. Ist $E \equiv C$, so ist $E(t)$ nach Konstruktion unserer Interpretation erfüllt. Ist $E \equiv (\neg C)(t)$, so kann $C(t)$ nicht auf dem Pfad vorkommen, da wir sonst im Tableau-Verfahren doch einen Widerspruch gefolgert hätten. Nach Modellkonstruktion ist damit $t \notin C^{\mathcal{I}}$, also $t \in \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} = (\neg C)^{\mathcal{I}}$.

Ist $E \equiv C \sqcap D$, so müssen sich bereits $C(t)$ und $D(t)$ auf dem Pfad befinden, da wir sonst $\to_\sqcap$ anwenden könnten. Nach Induktionsvoraussetzung ist $t \in C^{\mathcal{I}}, D^{\mathcal{I}}$, also auch $t \in C^{\mathcal{I}} \cap D^{\mathcal{I}} = (C \sqcap D)^{\mathcal{I}}$.

Ist $E \equiv C \sqcup D$, so muss entsprechend der Bedingung von $\to_\sqcup$ sich $C(t)$ oder $D(t)$ auf dem Pfad befinden. Aus $t \in C^{\mathcal{I}}$ oder $t \in D^{\mathcal{I}}$ folgt $t \in C^{\mathcal{I}} \cup D^{\mathcal{I}} = (C \sqcup D)^{\mathcal{I}}$.

Ist $E \equiv \forall R.C$, so sagt die Bedingung von $\to_\forall$, dass für alle u mit $R(t, u)$, also $(t, u) \in R^{\mathcal{I}}$, sich auch $C(u)$ auf dem Pfad befindet, nach Induktionsannahme also $u \in C^{\mathcal{I}}$. Es folgt $t \in \{r \mid \forall s.(r, s) \in R^{\mathcal{I}} \to s \in C^{\mathcal{I}}\} = (\forall R.C)^{\mathcal{I}}$.

Ist $E \equiv \exists R.C$, so sagt die Bedingung von $\to_\exists$, dass es ein u mit $R(t, u)$, also $(t, u) \in R^{\mathcal{I}}$, und $C(u)$ auf dem Pfad gibt. Nach Induktionsannahme ist $u \in C^{\mathcal{I}}$, also $t \in \{r \mid \exists s.(r, s) \in R^{\mathcal{I}}, s \in C^{\mathcal{I}}\} = (\exists R.C)^{\mathcal{I}}$.

Entscheiden wir nun noch die Allgemeingültigkeit von $D \sqsubseteq C$, also die Unerfüllbarkeit von

$D \sqcap \neg C$, in Negationsnormalform $E \equiv ((\forall R.A) \sqcup (\forall R.B)) \sqcap \exists R.(\neg A \sqcap \neg B)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\bot}{A(y)}\,(\neg A)(y)}{(\forall R.A)(x)}\,\to_\forall \quad \dfrac{\dfrac{\bot}{B(y)}\,(\neg B)(y)}{(\forall R.B)(x)}\,\to_\forall}{(\neg A)(y), (\neg B)(y)}\,\to_\sqcup}{R(x,y), (\neg A \sqcap \neg B)(y)}\,\to_\sqcap}{((\forall R.A) \sqcup (\forall R.B))(x), (\exists R.(\neg A \sqcap \neg B))(x)}\,\to_\exists}{(((\forall R.A) \sqcup (\forall R.B)) \sqcap \exists R.(\neg A \sqcap \neg B))(x)}\,\to_\sqcap$$

Zusammenfassend haben wir komplett algorithmisch entschieden, dass $(\forall R.A) \sqcup (\forall R.B) \sqsubseteq \forall R.(A \sqcup B)$, aber nicht $\forall R.(A \sqcup B) \not\sqsubseteq (\forall R.A) \sqcup (\forall R.B)$, wofür uns der Algorithmus auch ein Gegenbeispiel konstruiert hat.

Ein weiteres Beispiel befindet sich auf dem Handout zum Vortrag.

Erweiterung auf $\mathcal{ALCN}$ Wir können den Tableau-Algorithmus leicht auf die Erweiterung $\mathcal{ALCN}$ anpassend. Dabei spielt nun eine Rolle, ob in einer ABox vorkommende Individuennamen identisch sein können. Die sonst übliche *Unique Name Assumption* verwerfen wir und führen stattdessen explizite syntaktische Ungleichheitskonstrukte der Form $x \dot{\neq} y$ ein. Diese sollen semantisch genau dann als erfüllt gelten, wenn x und y in der Interpretation verschiedene Individuen bezeichnen. Für die neuen Konzeptkonstruktoren kommen folgende Regeln hinzu:

- Die $\to_{\geq}$-Regel
  - Bedingung: $\mathcal{A}$ enthält $(\geq nR)(x)$ und es gibt keine Individuennamen $z_1, \ldots, z_n$, so dass $R(x, z_i)$ für $1 \leq i \leq n$ und $z_i \dot{\neq} z_j$ für $1 \leq i < j \leq n$ in $\mathcal{A}$ liegen.
  - Aktion: $\mathcal{A} \to \mathcal{A} \cup \{R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \dot{\neq} y_j \mid 1 \leq i < j \leq n\}$ mit neuen Individuennamen $y_1, \ldots, y_n$.
- Die $\to_{\leq}$-Regel
  - Bedingung: $\mathcal{A}$ enthält $(\leq nR)(x)$ und es gibt Individuennamen $y_1, \ldots, y_{n+1}$, so dass $R(x, y_i)$ für $1 \leq i \leq n+1$ in $\mathcal{A}$ und $y_i \dot{\neq} y_j$ für gewisse $1 \leq i < j \leq n$ nicht in $\mathcal{A}$ liegt.
  - Aktion: $\mathcal{A} \to \{\mathcal{A}[y_i/y_j] \mid y_i \dot{\neq} y_j$ liegt nicht in $\mathcal{A}\}$

Hierbei steht $\mathcal{A}[y_i/y_j]$ für die ABox, die aus $\mathcal{A}$ durch Ersetzen aller Vorkommnisse von y_j durch y_i entsteht. Man verifiziert leicht, dass diese Regeln wieder die semantische Erhaltungseigenschaft erfüllen, dass die Erfüllheit einer ABox unter einer Interpretation $\mathcal{I}$ äquivalent ist zu der Erfülltheit einer der durch die Regelanwendung entstandenen ABoxen unter $\mathcal{I}$. Als zusätzliche Kollisionsbedingung für ABoxen $\mathcal{A}$ führen wir den Fall ein, dass $(\leq nR)(x) \in \mathcal{A}$ und $y_i \dot{\neq} y_j \in \mathcal{A}$ für Individuennamen $x, y_1, \ldots, y_{n+1}$ und einen Rollennamen R. Eine derartige ABox ist offensichtlich inkonsistent. Damit sieht man ebenfalls, wie sich die beiden neuen Regeln mit ihren Bedingungen in obigen Beweis der Vollständigkeit des Tableu-Algorithmus (ein vollständig derivierter, kollisionsfreier Pfad ergibt ein Modell der ursprünglichen ABox) einfügen. Man beachte, dass bei Anwendung der $\to_{\leq}$-Regel mit Parameter n quadratisch viele Alternativen zu betrachten sind mit möglicher Auswirkung auf die Komplexität (auch dazu später).

Terminierung

Als Beispiel für die neu eingeführten Regeln versuchen wir, die Erfüllbarkeit der ABox $\mathcal{A} = \{(\leq 1R)(x), (\exists R.A)(x), (\forall R.\exists R.A)(x), R(x,x)\}$ zu entscheiden. Die globale Individuennamensersetzungen der $\rightarrow_{\leq}$-Regel können wir nicht ohne weiteres in unsere inkrementelle Darstellungsweise übertragen. Statt pfadglobal y_j durch y_i zu ersetzen, werden wir die Gleichheit $y_j = y_i$ einfügen und im Folgenden den Variablennamen y_j vermeiden:

$$\begin{array}{c}
\dots \\ \hline
z = y \quad \rightarrow_{\forall} \\ \hline
R(z,u), A(u) \quad \rightarrow_{\leq} \\ \hline
(\exists R.A)(z) \quad \rightarrow_{\exists} \\ \hline
y = x \quad \rightarrow_{\forall} \\ \hline
R(y,z), A(z) \quad \rightarrow_{\leq} \\ \hline
(\exists R.A)(y) \quad \rightarrow_{\exists} \\ \hline
R(x,y), A(y) \quad \rightarrow_{\forall} \\ \hline
(\leq 1R)(x), (\exists R.A)(x), (\forall R.\exists R.A)(x), R(x,x) \quad \rightarrow_{\exists}
\end{array}$$

Durch weitere periodische Anwendung derselben Regeln werden wir den Pfad nie zu Ende führen: Die ABox in der obersten Zeile ist isomorph zu der ABox drei Zeilen unter ihr: Effektiv wurde lediglich der Individuenname z durch u ersetzt. Eine andere Reihenfolge der Regelanwendung führt zum Ziel:

$$\begin{array}{c}
y = x \quad \rightarrow_{\leq} \\ \hline
R(x,y), A(y) \quad \rightarrow_{\exists} \\ \hline
(\leq 1R)(x), (\exists R.A)(x), (\forall R.\exists R.A)(x), R(x,x)
\end{array}$$

In der resultierenden ABox $\{(\leq 1R)(x), (\exists R.A)(x), (\forall R.\exists R.A)(x), R(x,x), A(x)\}$ können wir keine weiteren Regeln anwenden (und damit eine Interpretation ablesen: $\Delta^{\mathcal{I}} = \{x\}, A^{\mathcal{I}} = \{x\}, R^{\mathcal{I}} = \{x,x\}$).

Analysieren wir, wie es zu der Nichtterminierung im ersten Versuch kommen konnte, so stellen wir fest, dass die Unterformel-Eigenschaft des Tableau-Algorithmus uns zwar garantiert, dass nur eine endliche Menge an Konzeptformeln generiert wird (nämlich die, die auch schon als Unterformeln der ursprünglichen ABox auftreten), für die auftretenden Individuennamen jedoch keine Beschränkung gegeben wird, so dass prinzipiell unendlich viele Instanzen desselben Konzeptes mit verschiedenen Individuennamen generiert werden können wie im vorletzten Beispiel.

Ohne die $\rightarrow_{\leq}$-Regel, d. h. für ABoxen ohne Formeln mit Konstrukten der Form $\leq nR$, ist die Zahl an Individuennamen beschränkt und Terminierung daher immer gegeben. Wechselseitig induktiv können wir nämlich zeigen: Für jeden in einer ABox $\mathcal{A}$ im Tableau vorkommenden Individuennamen z gibt es eine minimal lange Folge $z_0, \dots, z_k = z$ mit einer Folge von Rollennamen $R_1, \dots, R_l$, so dass $R_1(z_0, z_1), \dots, R_l(z_{l-1}, z_l) \subseteq \mathcal{A}$ und z_0 ein in der ursprünglichen ABox vorkommender Individuenname ist. Wir sagen, dass z auf Level l in $\mathcal{A}$ auftaucht. Für $C(z) \in \mathcal{A}$ und z auf Level l ist die maximale Rollentiefe, also die maximale Anzahl ineinander verschachteter Rollenkonstruktoren, von C beschränkt durch die maximale Rollentiefe aller Formeln der ursprünglichen ABox minus l. Schließlich ist für jeden Individuennamen ist die Anzahl der Rollennachfolger in $\mathcal{A}$ beschränkt durch die Summe der Zahlen n in Konstrukten der Form $\geq nR$ zusammen mit der Zahl von Konstruktoren der Form $\exists R.C$ in Formeln der ursprünglichen ABox.

Mit der $\rightarrow_{\leq}$-Regel wird die Sache komplizierter, da ihre Umbennenungsaktion eine gewisse Nonlokalität induziert. Für ABoxen der Form $\{C(x)\}$, wie sie bei Subsumptions- und Konsistenz-

problemen von Konzepten auf dem Terminologie-Level auftreten, funktioniert obiger Beweis jedoch noch im Wortlaut. Er zeigt außerdem, dass $\mathcal{ALCN}$ die *endliches-Baummodell-Eigenschaft*[3] hat, d. h. jedes erfüllbare Konzept durch eine endliche Interpretation erfüllbar ist, deren Rollenrelationen als gerichteten Graphen einen Wurzelbaum ergeben. Für allgemeine ABoxen, die insbesondere auch Rollenbeschreibungen enthalten können, kann man zeigen, dass Terminierung gegeben ist, wenn wir Regeln, die neue Individuennamen generieren, also die $\rightarrow_\exists$- und $\rightarrow_\geq$-Regeln erst dann anwenden, wenn keine anderen Regeln mehr anwendbar sind. Eine andere Möglichkeit, Terminierung wiederzugewinnen ist die folgende: In einem ersten Schritt wenden wir die Transformationsregeln nur auf Formeln der Form $C(x)$ an, in denen x ein Individuenname ist, der bereits in der ursprünglichen ABox auftrat. An diesem Punkt haben die in der ABox enthaltenen Rollenbeschreibungen ihre Arbeit erledigt und werden uns keine neuen Deduktionen mehr ermöglichen. Wir brauchen sie folglich nicht mehr zu betrachten und können separat für jeden Individuennamen x mit vorkommenden Formeln $C_1(x), \ldots, C_k(x)$ eine Instanz des Tableau-Algorithmus auf die ABox $\{(C_1 \sqcap \ldots \sqcap C_k)(x)\}$ spezieller obiger Form anwenden.

Komplexität

Beschränken wir uns im Folgenden auf ABoxen der Form $\{C(x)\}$, also das Problem der Konzepterfüllbarkeit. Dem Beweis der Terminierung des Tableau-Verfahrens für diese Form der Ausgangs-ABox folgend sehen wir, dass dabei prinizipiell exponentiell große Strukturen entstehen können bzw. durchlaufen werden müssen. Versuchen wir zunächst, einen effizienten nichtdeterministischen Algorithmus zu konstruieren.

In einem nichtdeterministischen Algorithmus können wir uns bei Anwendung einer der Regeln $\rightarrow_\sqcup$ und $\rightarrow_\leq$ auf eine Alternative beschränken. Wenden wir stets mit höchster Priorität die für die Individuennamen neutralen Regeln $\sqcap$ und $\sqcup$ an und generieren für einen Individuennamen x mittels der $\rightarrow_\exists$- und $\rightarrow_\geq$-Regeln zunächst alle direkten Nachfolger bevor wir uns diesen zuwenden, so können wir die wegen Instanzen der $\leq$-Regel notwendigen Identifikationen der Nachfolger von x an dieser Stelle anwenden ohne uns später darum kümmern zu müssen. [4] Dies erlaubt uns, im Folgenden nichtdeterministisch einen einzigen Nachfolger und den zugehörigen Unterbaum im Baummodell zu betrachten, so dass unser Algorithmus letztendlich nur einen einigen Pfad im Baummodell generiert. Da die Länge dieses Pfades (also die Zahl der erzeugten Individuennamen) durch die Rollenkonstruktorverschachtelungstiefe und die Zahl der auf dem Pfad auftretenden Formeln durch die Zahl der Unterformeln des Ausgangskonzeptes beschränkt, hat dieser nichtdeterministische Algorithmus einen Speicherbedarf polynomiell in der syntaktischen Länge der Eingabeformel, liegt also in `NPSPACE`.

Nach einem bekannten Resultat von Savitch ist `NPSPACE` = `PSPACE`. Tatsächlich kann man zeigen, dass bereits die Erfüllbarkeit von $\mathcal{ACL}$-Konzepten `PSPACE`-hart ist.

Ausblick

Der vorgestellte Algorithmus ist ein kompaktes Beispiel für eine große Reihe Tableau-basierter Erfüllbarkeitsalgorithmen, die von sehr speziellen zu sehr allgemeinen Logiken reichen. Je nach Grad

[3] for lack of a better word

[4] Wenn die numerischen Parameter in den Konstrukten der Form $\geq nR$ bzw. $\leq nR$ nicht unär, also z. B. wie gewohnt binär, kodiert werden, können wir es uns nicht leisten, durch Invokationen der $\geq nR$-Regel tatsächlich n Individuennamen einzuführen - stattdessen führen wir einen neuen Kollisionstyp der Form $(\leq nR)(y), (\geq mR)(y) \in \mathcal{A}$ für $n < m$ ein.

der Erweiterung sind es zum Teil nicht einmal mehr Entscheidungsverfahren, wie beispielsweise das bekannte Tableau-Verfahren für die volle Prädikatenlogik (denn die Menge der wahren prädikatenlogischen Formeln ist unentscheidbar). Wir können uns fragen, wie weit wir unseren Algorithmus erweitern können, so dass seine theoretische Komplexität nicht wächst bzw. er noch garantiert terminiert, d. h. ein Entscheidungsverfahren bleibt.

Eine geringe Erweiterung besteht darin, syntaktisch auch qualifizierte Zahlenbeschränkungen für Rollennachfolger (Ausdrücke der Form $\leq nR.C$ bzw. $\geq nR.C$ zuzulassen und den Algorithmus entsprechend anzupassen. Es stellt sich heraus, dass diese Erweiterung immer noch in `PSPACE` liegt.

Definitorische (azyklische) TBoxen können wir behandeln, indem wir definierte Konzepte rekursiv durch die rechten Seiten in ihren Definitionen ersetzen. Dadurch können Konstrukte exponentieller Länge entstehen, deren Erfüllbarkeit im Allgemeinen (natürlich ohne diese Expansion) nicht mehr (gemessen an der syntaktischen Länge der TBox und der Eingabeformel) in `PSPACE` zu entscheiden ist.

Die Zulassung allgemeiner Inklusionsaxiomen bringt mehr Komplikationen mit sich. Mehrere Axiome der Form $C_1 \sqsubseteq D_1, \ldots, C_k \sqsubseteq D_k$ können wir äequivalent umformen zu $\top \sqsubseteq (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_k \sqcup D_k)$. Im Algorithmus führen wir dann für jeden neu eingeführten Individuennamen eine Instanz der Formel auf der rechten Seite ein. Dabei Terminierung nicht zu verlieren stellt sich als trickreich heraus, ist aber zu schaffen mittels gewisser Blockaderegeln, die weitere Bedingungen für die Anwendung gewisser Transformationsregeln schaffen (siehe auch den Vortrag zum Tableau-Beweiser für $\mathcal{SHOIN}$). Selbst bei Beschränkung auf einziges Inklusionsaxiom steigt die Komplexität und das Problem wird `EXPTIME`-vollständig.

Bekannte Komplexitätsklassen von Erweiterungen mittels Rollenkonstruktoren: Das Einführen eines Schnittoperators für Rollen lässt die Komplexität gleich, sofern die numerischen Parameter in den $\leq$- und $\geq$-Konstrukten unär angegeben wird. Ein Komplementoperator macht das Entscheidungsproblem `EXTTIME`-vollständig, beide zusammen ergeben sogar `NEXTTIME`-Vollständigkeit. Transitiver Abschluss, Verkettung, Vereinigung und Inversion ergeben zusammen `EXPTIME`-vollständige Subsumptions- und Erfüllbarkeitsprobleme.

Literatur

[1] Frank Baader and Werner Nutt. Basic description logics. In *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.