

Seminararbeit zum Thema

Hybride Integration von Description Logics und Datalog[∇] nach Rosati 2005

Michael Meier

Prof. Dr. G. Lausen
Betreuer

Prof. Dr. G. Lausen
Lehrstuhl für Datenbanken und Informationssysteme
Albert-Ludwigs-Universität Freiburg i. Br.

18. Juli 2006

Inhaltsverzeichnis

1	Einführung	4
1.1	Motivation	4
1.2	Kollision zweier Welten	4
1.2.1	Open World vs. Closed World	4
1.2.2	non-Unique Names vs. Unique Names	5
1.2.3	Theorie und Praxis	6
2	r-hybride Wissensbasen	6
2.1	Grundsätzliches	7
2.2	Syntax r-hybrider Wissensbasen	8
2.3	Semantik r-hybrider Wissensbasen	9
3	Diskussion	14
3.1	Komplexität und Entscheidbarkeit	14
3.2	Wie wurden die Integrationsprobleme gelöst?	15
4	Fazit	16
	Literatur	16

1 Einführung

Ich stelle einen neuen Ansatz zur hybriden Integration von Description Logics und Datalog vor. Diese Seminararbeit basiert auf den Arbeiten [Ros05a] und [Ros05b], wobei ich die zweite Arbeit, der einfacheren Notation halber, nicht betrachte.

1.1 Motivation

Datalog und Description Logics (DL) sind zwei wohlbekannte Ansätze zur Verarbeitung von Wissen in Datenbanken. Datalog ist ein regelbasierter Ansatz, was es, je nach Semantik, möglich macht, eben diese Regeln effizient auszuwerten, unter gewissen Restriktionen an die Ausdrucksstärke der verwendeten Logik. Description Logics sind syntaktische Varianten von Fragmenten der Logik der ersten Stufe. Ihr Vorteil besteht vor allem darin Ontologien in die Wissensverarbeitung miteinbeziehen zu können. Die Komplexität liegt oft in NEXPTIME oder ist gar unentscheidbar.

Warum sollte man also nicht versuchen beide Welten miteinander zu verbinden um gewisse Synergieeffekte zu erhalten? Eine hybride Integration gestaltet sich jedoch schwierig, weil Description Logics und Datalog von gegensätzlichen Annahmen ausgehen!

Zwei bestehende Ansätze zur hybriden Integration sind CARIN und AL-LOG. Beide Ansätze wurden nur für spezielle Description Logics (z.B. ALC) entwickelt und haben die Einschränkung, dass die DL-Komponente zwar die Datalog-Komponente "beeinflussen" kann, aber nicht umgekehrt. Unter Beeinflussung ist hierbei folgendes zu verstehen: Regeln in der Datalog-Komponente dürfen Relationssymbole aus der DL-Komponente enthalten, d.h. die DL-Komponente hat in diesem Sinne einen Einfluss auf die Datalog-Komponente, jedoch dürfen durch Datalog-Regeln keine DL-Prädikate hergeleitet werden. In diesem Sinne ist die Interaktion beider Komponenten einseitig beschränkt. Zumindest diese Restriktionen werden durch den Ansatz von Rosati aus dem Jahre 2005 aufgehoben.

1.2 Kollision zweier Welten

Bevor ich im nächsten Kapitel den Ansatz konkret vorstelle, möchte ich hier zunächst die grundsätzlichen Probleme der Integration von Datalog und DL darstellen.

1.2.1 Open World vs. Closed World

Dies scheint der schwierigste Punkt überhaupt bei der Integration zu sein. Die grundlegende Fragestellung ist die folgende: Wie gehe ich mit Daten um, welche nicht in meiner Datenbank vorhanden sind und auch nicht hergeleitet werden können? Die Antwort auf diese Frage hängt von meiner Anwendung ab. Ein Beispiel: ein Roboter fährt über das Freiburger Campus-Gelände und sammelt Daten über seine Umgebung. Wenn man nun die Anfrage stellt, ob es in Sydney gerade regnet, würde man wohl eher die Antwort "Ich weiß es nicht." erwarten als "Nein, es regnet nicht." . Wenn man im Gegensatz dazu in einer Datenbank eine Kundenliste eines Unternehmens gespeichert hat und man stellt

nach einer Anfrage fest, dass die Firma Müller darin nicht auftaucht, wird man wohl schließen, dass die Firma Müller kein Kunde ist und man wird nicht schließen, dass man nicht weiß, ob sie Kunde ist oder nicht. Also, je nach Art der Anwendung werden nicht herleitbare Daten als falsch oder als nicht ableitbar angenommen, letzteres entspricht einem "Ich weiß es nicht."

Die Open World Assumption entspricht dabei der Annahme der klassischen Logik, dass es nicht ableitbare Fakten geben kann (siehe z.B. die Gödelschen Unvollständigkeitssätze). Die Closed World Assumption suggeriert die Vollständigkeit des Datenbestandes in einer Wissensbasis: alles was nicht ableitbar ist, ist falsch. Dies klingt simpel hat aber große Konsequenzen!

Beispiel 1.1 (Was Closed World bedeutet)

Der folgende Code drückt keinerlei Präferenz für eines seiner beiden minimalen Modelle aus:

$$\begin{aligned} & \text{boy}(X) \vee \text{girl}(X) \leftarrow \text{PERSON}(X) \\ & \text{PERSON}(\text{bob}) \end{aligned}$$

Hier ist das anders:

$$\begin{aligned} & \text{boy}(X) \leftarrow \text{PERSON}(X), \text{ not } \text{girl}(X) \\ & \text{PERSON}(\text{bob}) \end{aligned}$$

$\text{girl}(\text{bob})$ ist nicht herleitbar, also schlägt die Closed World Assumption zu und $\text{girl}(\text{bob})$ wird als falsch angenommen, d.h. hier gibt es nur ein einziges minimales Modell.

Wenn man obige Ausdrücke unter der erststufigen Semantik betrachtete, wären sie logisch äquivalent (offensichtliche syntaktische Transformationen). Dies zeigt, dass die Frage ob man Closed oder Open World Assumption nutzt, große Auswirkungen auf das Resultat einer Auswertung hat.

□

Wenn man also DL und Datalog unter Nutzung ihrer bestehenden Semantiken hybride integrieren will, muss man in irgendeiner Art und Weise Closed und Open World Assumption zusammenbringen.

1.2.2 non-Unique Names vs. Unique Names

In einer DL kann es leicht vorkommen, dass syntaktisch verschiedene Objekte semantisch gleichgesetzt werden (non-Unique Names Assumption). Unique Names bedeutet gerade, dass dies nicht passiert. Datalog verwendet die Unique Names Assumption, einfach aus dem Grund, dass es in Datalog, im Gegensatz zu einer DL, zunächst einmal kein Gleichheitsprädikat gibt.

Beispiel 1.2 (Was Unique Names bedeutet)

$C := (\exists \leq 1 \text{ hasChild})$
 $C(\text{mary})$
 $\text{hasChild}(\text{mary}, \text{bob})$
 $\text{hasChild}(\text{mary}, \text{alice})$

Offensichtlich werden hier *bob* und *alice* semantisch gleichgesetzt, obwohl sie syntaktisch verschieden sind.

□

1.2.3 Theorie und Praxis

Was passiert bei der Integration mit der Entscheidbarkeit? Bleibt sie immer erhalten oder wird das Gesamtproblem unentscheidbar? Was passiert ggf. mit der Komplexität? Explodiert sie? Oder gibt es nur einen sanften Anstieg? Diese Fragen kann man als Theoretiker stellen. Von praktischer Seite relevant ist da eher die Frage, ob man ein komplett neues System entwickeln muss, oder ob man bestehende Implementierungen wiederverwenden kann. Ideal wäre eine Architektur bei der nur ein zusätzliches Modul geschrieben werden muss zur Kommunikation mit dem DL- und Datalog-Reasoner, z.B.

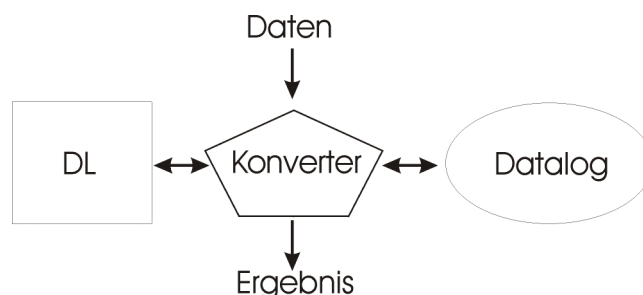


Abbildung 1: Mögliche Architektur eines hybriden Systems

Wenn der Zusatzaufwand im Konverter klein gehalten werden könnte, dann wäre ein solches System durchaus realistisch.

2 r-hybride Wissensbasen

Nun soll der eigentlich Ansatz vorgestellt werden. Zuerst diskutiere ich Syntax und dann in der Semantik Modell- und Folgerungsbeziehung.

Der Vorschlag von Rosati stammt aus dem Jahre 2005. Er basiert auf zwei Komponenten: einer strukturellen Komponente für eine DL und einer Datalog[∇]-Komponente.

Es wird im Wesentlichen so sein, dass für beide Komponenten eine bereits existierende Semantik verwendet wird und die eigentliche Integrationsleistung darin besteht, dafür zu sorgen, dass beide Komponenten keine widersprüchlichen Informationen liefern. Die strukturelle Komponente ist dabei nicht an eine spezifische DL wie z.B. OWL-DL oder ALC gebunden, sondern verwendet die Syntax der Sprache der ersten Stufe. DLs sind in ihrer Mächtigkeit eingeschränkte syntaktische Varianten der Sprache der ersten Stufe, insbesondere verwenden sie die Semantik der Sprache der ersten Stufe. Die Einschränkungen an die Syntax, haben insofern Auswirkungen auf das Reasoning, als dass Reasoning-Algorithmen nicht stur auf Resolution beruhen müssen, sondern dass teilweise effizientere Algorithmen ausreichen um Korrektheit und Vollständigkeit des Reasonings zu erreichen. Insofern sind DLs interessant! DL-Syntax übersetzt sich in natürlicher Art und Weise in die erste Stufe.

Beispiel 2.3 (DL \rightarrow FOL)

DL	FOL
$C \sqsubseteq D$	$\forall x(Cx \rightarrow Dx)$
$D \equiv \forall R.C$	$\forall x(Dx \leftrightarrow \forall y(Rxy \rightarrow Cy))$
$D \sqsubseteq \exists R.C$	$\forall x(Dx \rightarrow \exists y(Rxy \wedge Cy))$

Dabei sind C, D primitive Konzepte und R ein primitiver Rollenname. Auf geeignete Symbolmengen, so dass obige Ausdrücke überhaupt formulierbar sind, ist unbedingt zu achten. Für eine ausführlichere Diskussion über die Beziehung zu anderen Formalismen, verweise ich auf das DL-Handbook.

□

Auf der Datalog[∇]-Seite verwendet Rosati stabile Modelle, was den simplen Grund hat, dass möglichst viele Datalog-Programme ausführbar sein sollen, basierend auf einer zwei-wertigen Logik. Der Einfachheit halber möchte ich im Folgenden nur die Semantik stratifizierbarer Datalog[∇]-Programme ohne Disjunktion im Kopf verwenden. Syntaktisch gesehen folge ich aber Rosati vollständig. Demjenigen Leser der der Schönheit der stabilen Semantik erlegen ist, empfehle ich die Original-Arbeit Rosatis wärmstens.

2.1 Grundsätzliches

Für den Rest dieses Papiers legen wir folgende Konventionen bezüglich der zugrundeliegenden Sprache fest. Sei $\mathcal{A} := \mathcal{A}_S \cup \mathcal{A}_R$ mit $\mathcal{A}_S \cap \mathcal{A}_R = \emptyset$ eine endliche Menge von Prädikatsymbolen. $P \in \mathcal{A}_S$ heißt strukturelles Prädikat, es wird immer groß geschrieben. Da solch ein P in der strukturellen (DL) Komponente auftauchen wird, muss es notwendigerweise ein- oder zweistellig sein. Für Prädikate aus \mathcal{A}_S gibt es keine Beschränkung für die Stelligkeit. \mathcal{C} sei eine endliche Menge von Konstantensymbolen. Sei $\mathcal{L} \subseteq L_0^{\mathcal{A} \cup \mathcal{C}}$ (Notation Prof. Flum), d.h. \mathcal{L} ist eine Menge von Sätzen einer Sprache der ersten Stufe ohne Funktionssymbole. Man stelle sich der Einfachheit halber vor, dass \mathcal{L}

durch ein Kalkül, wie bei DLs üblich, gegeben ist. Intuitiv, stellt \mathcal{L} also diejenige Sprache dar, in welcher ich in der strukturellen Komponente Ausdrücke formulieren darf.

Beispiel 2.4 (Beispiel einer r-hybriden Wissensbasis)

strukturelle Komponente:

$\forall x(PERSON(x) \rightarrow BEING(x))$
 $\forall x(FEMALE(x) \rightarrow \neg MALE(x))$
 $PERSON(bob)$
 $FEMALE(mary)$

Datalog^{¬∨}-Komponente:

$boy(X) \leftarrow PERSON(X), enrolled(X), \text{not } girl(X)$
 $FEMALE(X) \leftarrow girl(X)$
 $MALE(X) \leftarrow boy(X)$
 $enrolled(bob)$

Die Symbole unserer Sprache können beispielsweise wie folgt gewählt werden:

$\mathcal{A}_S = \{FEMALE, MALE, PERSON, BEING\},$
 $\mathcal{A}_R = \{girl, boy, enrolled\},$
 $\mathcal{C} = \{bob, mary\}$

Dabei sei \mathcal{L} so mächtig, dass obige Ausdrücke formulierbar sind.

□

2.2 Syntax r-hybrider Wissensbasen

Nun machen wir das obige Beispiel ganz offiziell zur r-hybriden Wissensbasis.

Definition 2.5 (r-hybride Wissensbasis)

$\mathcal{H} = (\mathcal{T}, \mathcal{P})$ heißt **r-hybride Wissensbasis** genau dann wenn

- $\mathcal{T} \subseteq \mathcal{L}$, \mathcal{T} endlich
- kein Prädikat aus \mathcal{A}_R kommt in \mathcal{T} vor
- \mathcal{P} ist ein Datalog^{¬∨}-Programm mit Prädikaten aus \mathcal{A} und Konstanten aus \mathcal{C}
- keine Regel aus \mathcal{P} darf negierte Prädikate aus \mathcal{A}_S enthalten
- **safeness:** in einer Regel muss jede Variable, die darin vorkommt, auch in einem positiven Prädikat aus \mathcal{A}_R vorkommen

Zur Erinnerung nochmals die Definition der Syntax von Datalog^{¬∨}-Programmen.

Definition 2.6 (Datalog^{¬∨}-Programm)

Ein Datalog^{¬∨}-Programm besteht Regeln der Form:

$$p_1(X_1) \vee \dots \vee p_n(X_n) \leftarrow r_1(Y_1), \dots, r_m(Y_m), \text{ not } u_1(W_1), \dots, \text{ not } u_h(W_h)$$

wobei die X_i, Y_i, W_i Tupel von Variablen und Konstanten sind und $n, m, h \geq 0$ gilt.

Man verifiziere in Beispiel 2.4, dass es sich tatsächlich um eine r-hybride Wissensbasis handelt und störe sich dabei nicht daran, dass \mathcal{L} nicht explizit gewählt wurde. Da es mindestens genauso wichtig ist zu wissen, was man nicht darf, hier noch ein

Beispiel 2.7 (Nicht-Beispiel einer r-hybriden Wissensbasis)

strukturelle Komponente:

$\forall x(PERSON(x) \rightarrow BEING(x))$
 $\forall x(FEMALE(x) \rightarrow \neg MALE(x))$
 $PERSON(bob)$
 $FEMALE(mary)$
 $girl(mary)$

Datalog^{¬∨}-Komponente:

$boy(X) \leftarrow PERSON(X), \text{ not } girl(X)$
 $FEMALE(X) \leftarrow \underline{girl(X)}$
 $MALE(X) \leftarrow boy(X), \underline{\text{ not } FEMALE(X)}$

falls $\mathcal{A}_S = \{FEMALE, MALE, PERSON, BEING\}$, $\mathcal{A}_R = \{girl, boy, enrolled\}$,
 $\mathcal{C} = \{bob, mary\}$.

Ich überlasse es dem Leser sich klarzumachen, dass die drei wesentlichen syntaktischen Restriktionen in der Definition der r-hybriden Wissensbasis durch die drei unterstrichenen Stellen im obigen Programm verletzt werden.

□

2.3 Semantik r-hybrider Wissensbasen

Was ist die Semantik der durch die Definition der r-hybriden Wissensbasis erklärten Zeichenreihen? Was würde man intuitiv erwarten, was passieren soll in der Semantik? Wozu sind die angegebenen syntaktischen Restriktionen da? Wieso überhaupt solch starke Restriktionen?

Zur Beantwortung der letzten Frage verweist Rosati auf CARIN. Wenn man zuviel Interaktion zwischen struktureller und Datalog-Komponente zulässt wird die Erfüllbarkeit eines solchen Systems unentscheidbar. Wozu sind die angegebenen syntaktischen Restriktionen da? Die safeness-Bedingung hat starke Auswirkungen auf die Semantik. Rosati hat die Semantik völlig anders definiert als das im Folgenden passieren wird. Die safeness-Bedingung ermöglicht es erst, dass ich die Definition der Semantik so formulieren kann, wie ich das tun werde. Die syntaktische Einschränkung, die da lautet “ keine Regel aus \mathcal{P} darf negierte Prädikate aus \mathcal{A}_S enthalten“ wird hier leider nicht ersichtlich werden.

Die Semantik von \mathcal{H} wird über eine Herbrand-Interpretation \mathcal{I} gegeben sein. Für die strukturelle Komponente nehmen wir hier die Unique Name Assumption an. Wir werden uns jetzt zunächst auf anschauliche Art und Weise das Recht erarbeiten ein Modell von \mathcal{T} als Herbrandinterpretation, d.h. als Menge von zutreffenden Prädikaten schreiben zu dürfen. Ein Modell von \mathcal{T} ist zunächst einmal eine Interpretation $\mathcal{I} = (\mathcal{A}, \beta)$, wobei \mathcal{A} eine Struktur (zur richtigen Symbolmenge) ist und β eine Belegungsfunktion für die freien Variablen. Trivialerweise kommen aber gar keine freien Variablen vor, also ist β irrelevant. Bleibt also die Struktur \mathcal{A} . Sei $\mathcal{A} = (A, R_1^A, \dots, R_n^A, c_1^A, \dots, c_m^A)$, wobei A der Träger der Struktur, die R_i Relationssymbole und die c_i Konstantensymbole sind. Die Unique Name Assumption impliziert, dass $c_i^A = c_i$ gilt. Also brauchen wir die Interpretation der Konstanten nicht explizit in unsere Struktur hineinzuschreiben. Als Träger setzen wir $A = \mathcal{C}$, demnach brauchen wir den Träger auch nicht mehr explizit aufzuführen. Bleibt die Interpretation der Relationssymbole. Hier denkt man sich einfach, dass die Relationen durch vollständige Enumerierung der in Relation stehenden Tupel (Konstantensymbole) gegeben sind. Schon wurde aus einer erststufigen Interpretation eine Menge von Prädikaten, also so etwas wie ein Herbrandmodell. Also können wir uns für \mathcal{T} und \mathcal{P} Herbrandmodelle basteln und darauf dann die üblichen Mengenoperationen anwenden um eine Semantik für \mathcal{H} zu erhalten.

Für ein Herbrandmodell \mathcal{I} ist also die Bedeutung von $\mathcal{I} \models_{Ro} \mathcal{H}$ zu definieren, also die Modellbeziehung. In Wirklichkeit sind wir aber nicht an der Modellbeziehung interessiert, sondern an der Folgerungsbeziehung, also an $\mathcal{H} \models_{Ro} \varphi$ (φ z.B. ein existentieller Ausdruck). Diese wird dann in einem zweiten Schritt definiert.

Zunächst überlegen wir uns anhand des eingeführten Beispiels, was wir denn gerne hätten, was die Semantik leisten soll.

Beispiel 2.8 (Interpretation am Beispiel)

strukturelle Komponente:

$\forall x(PERSON(x) \rightarrow BEING(x))$

$\forall x(FEMALE(x) \rightarrow \neg MALE(x))$

$PERSON(bob) \quad FEMALE(mary)$

Datalog[¬]-Komponente:

$boy(X) \leftarrow PERSON(X), enrolled(X), not\ girl(X)$
 $FEMALE(X) \leftarrow girl(X)$
 $MALE(X) \leftarrow boy(X)$
 $enrolled(bob)$

$\mathcal{I} = \{boy(bob), enrolled(bob), MALE(bob), PERSON(bob), BEING(bob),$
 $FEMALE(mary), BEING(mary)\}$

Schön wäre es wenn ein Modell unserer Wissensbasis $PERSON(bob)$ und $enrolled(bob)$ nimmt und mit Hilfe der ersten Datalog-Regel und der Closed World Assumption den Fakt $boy(bob)$ herleitet. Aus $boy(bob)$ sollte dann mit Hilfe der dritten Datalog-Regel dann $MALE(bob)$ folgen und aus $MALE(bob)$ dann in der DL-Komponente $\neg FEMALE(bob)$. Man beachte, dass negative Literale in der Interpretation nicht explizit aufgeführt werden, sondern nur implizit, nämlich durch ihr dortiges Nicht-Erscheinen. Dies hat absolut nichts mit Closed oder Open World zu tun. Man rekapituliere, dass wir hier nur über die Modellbeziehung sprechen und nicht über die Folgerungsbeziehung. In der ersten Stufe erinnert dies an den einfachen Fakt, dass die Theorie einer Struktur vollständig ist. $BEING(bob)$ leitet sich im Beispiel auch leicht ab. Der einzige Fakt der nicht unmittelbar aus den Regeln/Ausdrücken folgt ist $BEING(mary)$. Dazu sage ich nochmals: wir reden über die Modellbeziehung und nicht über die Folgerungsbeziehung. Da $BEING(mary)$ zu keinen Widersprüchen führt, ist das in einem Modell also zulässig.

Nun splitten wir unsere Interpretation in zwei Teilinterpretation \mathcal{I}_S für die strukturelle Komponente und \mathcal{I}_R für die Datalog-Komponente.

$\mathcal{I}_S = \{MALE(bob), PERSON(bob), BEING(bob), FEMALE(mary), BEING(mary)\}$
 $\mathcal{I}_R = \{MALE(bob), PERSON(bob), boy(bob), enrolled(bob)\}$

Bezogen auf unsere beiden Komponenten sieht man, dass diese Teilinterpretationen jeweils alle Constraints erfüllen und dass beide Teilinterpretationen auf Prädikaten die potentiell in beiden Komponenten vorkommen können (hier: $MALE(bob)$, $PERSON(bob)$), übereinstimmen. D.h. egal ob man die strukturelle oder die Datalog-Komponente nach $MALE(bob)$ fragt, man erhält in beiden Fällen dieselbe Antwort. Hier spielt bereits die safeness-Bedingung mit hinein, wegen ihr brauche ich für die Interpretation des Datalog-Teils nur diejenigen Konstanten zu berücksichtigen, welche bereits in dem Datalog-Programm vorkommen. Bezogen auf das Beispiel bedeutet dies folgendes. Man könnte ja meinen, dass im DL-Teil $FEMALE(mary)$ gilt und im Datalog-Teil $\neg FEMALE(mary)$, weil $FEMALE(mary)$ nicht in \mathcal{I}_R auftaucht. Der Punkt ist, dass aufgrund der safeness-Bedingung $FEMALE(mary)$ niemals herleitbar wäre, weil die Konstante $mary$ nicht im Datalog-Teil auftaucht. Die Konstante $mary$ berührt somit den Datalog-Teil nicht und man lässt die Anfrage $FEMALE(mary)$ an die Datalog-Komponente erst gar nicht zu.

Insofern gibt es hier keinen Widerspruch.

□

Wenn man es schafft sich Interpretationen für die beiden Einzelkomponenten zu bauen, in der Art, dass diese sich nicht widersprechen, kann man beide Teilinterpretationen zu einer Gesamtinterpretation für die Wissensbasis vereinigen. Ein erster Schritt hierzu ist festzustellen, welche Prädikate potentiell in beiden Komponenten vorkommen können, um dies als Ausgangspunkt für eine Semantik des Gesamtsystems zu nehmen. Da dies auf rein syntaktischer Basis herausgefunden werden muss, ist dieser Schritt notwendigerweise sehr grob.

Zunächst einmal einige Hieroglyphen. Bezeichne $\mathcal{A}_S/\mathcal{P}$ die Menge der in \mathcal{P} vorkommenden Prädikaten-Symbole aus \mathcal{A}_S . $gr(\mathcal{P}) := \{m(t) : m \in \mathcal{A}_S/\mathcal{P}, m \text{ hat Stelligkeit } k \text{ und } t \text{ ist ein } k\text{-Tupel von Konstanten aus } \mathcal{P}\}$. $gr(\mathcal{P})$ entspricht denjenigen Prädikaten auf denen sich unsere beiden Komponenten potentiell in die Quere kommen können, in dem Sinne, dass eine Komponente sagt "Dieses Prädikat gilt!" und die andere Komponente sagt "Dieses Prädikat gilt nicht!". Sei (W, F) eine 2-Partition von $gr(\mathcal{P})$. Man mache sich klar, dass (W, F) einem potentiellen partiellen Modell von \mathcal{H} entspricht. Intuitiv gesprochen, nimmt man alle Elemente in W als wahr an und alle Elemente in F als falsch.

Beispiel 2.9 (Beispiel einer Partition)

strukturelle Komponente:

$\forall x(PERSON(x) \rightarrow BEING(x))$
 $\forall x(FEMALE(x) \rightarrow \neg MALE(x))$
 $PERSON(bob)$
 $FEMALE(mary)$

Datalog^{-V}-Komponente:

$boy(X) \leftarrow PERSON(X), enrolled(X), not\ girl(X)$
 $FEMALE(X) \leftarrow girl(X)$
 $MALE(X) \leftarrow boy(X)$
 $enrolled(bob)$

$gr(\mathcal{P}) = \{MALE(bob), PERSON(bob), FEMALE(bob)\}$
 $W = \{MALE(bob), PERSON(bob)\}$
 $F = \{FEMALE(bob)\}$

Die Elemente von $gr(\mathcal{P})$ ergeben sich direkt aus der Definition. W und F wurden im Prinzip willkürlich gewählt, bzw. so gewählt, dass am Ende auch wirklich ein Modell herauskommt.

□

Im nächsten Schritt fügt man die Elemente aus $gr(\mathcal{P})$ gemäß der gewählten Partition zu den beiden Komponenten hinzu.

Beispiel 2.10 (Beispiel einer Partition)

strukturelle Komponente:

$\forall x(PERSON(x) \rightarrow BEING(x))$
 $\forall x(FEMALE(x) \rightarrow \neg MALE(x))$
 $MALE(bob)$
 $PERSON(bob)$
 $\neg FEMALE(bob)$
 $FEMALE(mary)$

Datalog^{¬∨}-Komponente:

$boy(X) \leftarrow PERSON(X), enrolled(X), not\ girl(X)$
 $FEMALE(X) \leftarrow girl(X)$
 $MALE(X) \leftarrow boy(X)$
 $enrolled(bob)$
 $MALE(bob)$
 $PERSON(bob)$
 $\neg FEMALE(bob)$

$gr(\mathcal{P}) = \{MALE(bob), PERSON(bob), FEMALE(bob)\}$
 $W = \{MALE(bob), PERSON(bob)\}$
 $F = \{FEMALE(bob)\}$

Wenn sich nun auf den entstandenen Programmen Modelle definieren lassen, dann vereinigen wir diese und erhalten ein Modell für unsere gesamte Wissensbasis. Formal bedeutet dies folgende Definition der Modellbeziehung.

□

Definition 2.11 (Modellbeziehung)

$\mathcal{I}_S \cup \mathcal{I}_R \models_{Ro} \mathcal{H} :\Leftrightarrow$
es existiert (W, F) Partition von $gr(\mathcal{P})$ mit
 $\mathcal{I}_S \models_{FOL} \mathcal{T} \cup W \cup \neg F$ und $\mathcal{I}_R \models_{Datalog} \mathcal{P} \cup W \cup \neg F$

Definition 2.12 (Folgerungsbeziehung)

$\mathcal{H} \models_{Ro} \varphi :\Leftrightarrow$ für alle Interpretationen \mathcal{I} mit $\mathcal{I} \models_{Ro} \mathcal{H}$ gilt: \mathcal{I} erfüllt φ .

Man mache sich klar, dass gemäß dieser Definition aus unserer Beispielwissensbasis folgende positive Fakten folgen: $boy(bob)$, $MALE(bob)$, $BEING(bob)$, zusätzlich zu den Fakten, welche sowieso schon enthalten sind. Alle anderen positiven Prädikate folgen nicht.

Wir wissen, dass wenn die verwendete DL genügend reichhaltig ist, viele Reasoning Tasks auf die Erfüllbarkeit zurückgeführt werden können, deshalb geben wir hier Rosatis Algorithmus für die Erfüllbarkeit an.

Algorithmus R-Hybrid-SAT(\mathcal{H})

Input: r-hybride Wissensbasis $\mathcal{H} = (\mathcal{T}, \mathcal{P})$

Output: ja, wenn \mathcal{H} erfüllbar ist, nein sonst

begin

wenn es eine Partition (W, F) von $gr(\mathcal{P})$ gibt mit

(a) $\mathcal{T} \cup W \cup \{\neg r : r \in F\}$ hat ein Modell

(b) $\mathcal{P} \cup W \cup \{\neg r : r \in F\}$ hat ein stabiles Modell

dann gib ja zurück

sonst gib nein zurück

end

Wie finden wir eine geeignete Partition von $gr(\mathcal{P})$ effizient? Darauf gibt Rosati keine Antwort. Wir sind also zunächst einmal auf systematische Suche angewiesen, was diesen Algorithmus natürlich praktisch undurchführbar macht.

3 Diskussion

Wir diskutieren zunächst Fragen der Komplexität und Entscheidbarkeit und dann wie die genannten grundsätzlichen Probleme der Integration gelöst wurden.

3.1 Komplexität und Entscheidbarkeit

Die Erfüllbarkeit eines Datalog⁻-Programms \mathcal{P} unter der stabilen Semantik ist NEXPTIME-vollständig. Daraus folgt, dass die Erfüllbarkeit einer r-hybriden Wissensbasis (ohne Disjunktionen in \mathcal{P}) NEXPTIME-hart ist: erfülle (\emptyset, \mathcal{P}) und beachte dabei, dass die Semantik einer r-hybriden Wissensbasis in diesem Fall mit der stabilen Semantik übereinstimmt.

Für die Entscheidbarkeit erhalten wir ein, beweistechnisch gesehen, einfaches Ergebnis. Das Auffinden von Partitionen von $gr(\mathcal{P})$ ist per se entscheidbar. Dasselbe gilt für den Schritt (b) im Algorithmus. Demnach lautet unser Ergebnis wie folgt: wenn der Schritt (a) entscheidbar ist, dann ist auch das gesamte Erfüllbarkeitsproblem entscheidbar.

Verwendet man in der strukturellen Komponente $SHOIN(\mathcal{D})$ als DL, dann ist die Erfüllbarkeit von \mathcal{H} NEXPTIME-vollständig, also insbesondere entscheidbar.

Dies klingt alles ziemlich entmutigend, aber wenn man es schaffen könnte die Reasoning-Algorithmen für viele praktisch relevante Fälle so auszuprogrammieren, dass die Laufzeit in Grenzen gehalten werden kann, wären solche Systeme aber prinzipiell trotzdem einsetzbar.

3.2 Wie wurden die Integrationsprobleme gelöst?

Schauen wir uns zunächst an wie der Konflikt zwischen Open und Closed World Assumption gelöst wurde. Anschaulich betrachtet kann man sagen, dass wir auf den Prädikatensymbolen die ausschließlich in der strukturellen Komponente vorkommen von der Open World Assumption ausgehen und bei der Prädikaten aus \mathcal{A}_R von der Closed World Assumption. Dies folgt unmittelbar aus der Tatsache, dass die Reasoner für die jeweilige Komponente dort unabhängig voneinander agieren können. Interessanter wird das aber dann bei Prädikatensymbolen aus $\mathcal{A}_S/\mathcal{P}$. Hier könnte man sagen, dass wir durch die Definition der Modellbeziehung einen Teil der Closed World Assumption von der Datalog-Komponente in die strukturelle Komponente “hineinschieben”. Für die Folgerungsbeziehung bedeutet dies, dass wir auf Prädikaten aus $\mathcal{A}_S/\mathcal{P}$ unseren Datenbestand gerade soviel in wahr und falsch einteilen, wie uns abverlangt wird (durch alle möglichen Modelle) von der Datalog-Komponente.

Mit dem Gegensatz von non-Unique und Unique Name Assumption sind wir denkbar einfach umgegangen, denn wir haben ihn nicht betrachtet, das heißt wir sind von Unique Names ausgegangen. Dies ist zunächst eine Einschränkung, jedoch hat Rosati in [Ros05b] gezeigt, dass der Ansatz auch für non-Unique Name Assumption im Wesentlichen analog funktioniert.

Der Ansatz erhält die Entscheidbarkeit Erfüllbarkeit meistens, d.h. wenn die Erfüllbarkeit beider Einzelkomponenten entscheidbar sind, dann meistens auch die Erfüllbarkeit für das Gesamtproblem. Wir erhalten ein modulares Reasoning, welches es erlaubt auf bestehende Systeme zurückzugreifen unter relativ geringem Integrationsaufwand, relativ in dem Sinne, dass der Aufwand ein solches System auch wirklich zu bauen wesentlich schlimmer sein könnte. Wir haben einen bidirektionalen Informationsfluss zwischen beiden Komponenten: DL-Prädikate können in Köpfen und Rümpfen von Datalog-Regeln prinzipiell auftauchen, wenn auch teilweise eingeschränkt.

4 Fazit

Da der vorgestellte Ansatz nicht von einer konkreten DL abhängt, könnte man ihn als ein kleines bisschen allgemein bezeichnen. Es ist aber auch gerade diese Allgemeinheit an der wir den Komplexitätstod sterben. Hier bleibt noch viel zu tun, z.B. für speziellere Probleme einen effizienteren Algorithmus für die Erfüllbarkeit zu finden. Oder fallen wir etwa dem modularen Konzept des Ansatzes zum Opfer, weil wir die beiden Einzelsemantiken relativ direkt wiederverwenden? Würde ein Eingriff in die Reasoning-Algorithmen bessere Ergebnisse bringen? Man weiß es nicht, noch nicht.

Außerdem muss man sich auch immer die Frage stellen, ob die Semantik die wir erhalten auch in irgendeinem Sinne intuitiv ist. Ich möchte an dieser Stelle keine Diskussion über die Bedeutung des Wortes “intuitiv” anfangen, sondern möchte lediglich darauf hinweisen, dass ein hybrides System durch einen Anwender auch benutzbar sein muss. Als Anwender hat man gewisse Erwartungen daran, was die Zeichenreihen, die man in so ein System steckt, für ein Ergebnis bringen. Wenn die eigene Erwartung und die formal definierte Semantik sich nicht in gewissem Maße treffen, dann kann man so ein System wohl als unbenutzbar bezeichnen. Die vorgestellte Semantik scheint bei den wenigen überschaubaren Beispielen, welche ich mir selbst aufgeschrieben habe, gute Ergebnisse bezüglich der Intuitivität zu liefern, wobei damit natürlich nicht gesagt werden soll, dass damit die Semantik schon als intuitiv bezeichnet werden kann.

Hybride Integration spielt im Semantic Web eine wichtige Rolle. In [HPPSH05] wird die Befürchtung ausgesprochen, dass sich die Semantic Web-Gemeinde in zwei Lager aufspalten könnte, nämlich in Anhänger der Closed World und in Anhänger der Open World Assumption. Hybride Integration könnte das Semantic Web hier zumindest auf eine technologisch gemeinsame Grundlage stellen.

Literatur

- [HPPSH05] Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler, *Semantic web architecture: Stack or two towers?*, Principles and Practice of Semantic Web Reasoning (PPSWR 2005) (Francois Fages and Sylvain Soliman, eds.), LNCS, no. 3703, SV, 2005, pp. 37–41.
- [Ros05a] Riccardo Rosati, *On the decidability and complexity of integrating ontologies and rules*, Web Semantics **3** (2005), no. 1, 41–60.
- [Ros05b] Riccardo Rosati, *Semantic and computational advantages of the safe integration of ontologies and rules*, Proceedings of the Third International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2005), Lecture Notes in Computer Science, vol. 3703, Springer, 2005, pp. 50–64.