KI - Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

Anfragesprachen neu erdacht

François Bry

Received: date / Accepted: date

Zusammenfassung Zum Erfolg der relationalen Datenbanken haben ihre logik-basiereten Anfragen deswegen beigetragen, weil sie relativ leicht zu optimieren und auszudrücken sind. Relationale Anfragen eignen sich aber kaum zur Datenextraktion im Web und aus Ereignissströmen und können nicht so gut mit RDF umgehen. Dieser Artikel stellt prototypische Anfragesprachen vor, die das deklarative logikbasierte Paradigma der relationalen Anfragen auf verschiedene Weisen an das Web und an Ereignisströme anpassen: Xcerpt für unvollständige Web-Anfragen, KWQL für ein Spektrum von Web-Anfragetypen von einfachen Schlüsselwort-Anfragen bis hin zu Strukturanfragen, KWRL zur Begründungsverwaltung, KWRL zur Begründungsverwaltung, XChange^{EQ} für Ereignisanfragen, RDFLog zur uneingeschränkten Behandlung von Blank Nodes in RDF-Anfragen und RPL für Pfadanfragen in RDF-Graphen.

Schlüsselwörter Anfragesprachen · Visuelle Programmiersprachen · Complex Event Processing · Web · Semantic Web

1 Einleitung

Zum Erfolg der relationalen Datenbanken haben ihre logikbasierten, deklarativen Anfragesprachen (SQL in der Praxis, Datalog und Erweiterungen davon wie Datalog$^{\neg}$ in der Theorie) deswegen beigetragen, weil sie relativ leicht sowohl zu optimieren wie auch zu verwenden sind. Relationale Anfragen, seien sie in SQL oder in Erweiterungen von Datalog ausgedrückt, sind aber aus verschiedenen Gründen für Web-Informationssysteme wenig geeignet. Mit dem Web haben sich eben Informationssysteme in einer Weise grundlegend verändert, die Annahmen der relationalen Anfragesprachen widersprechen:

- Web-Informationssysteme sind meist heterogen in dem Sinne, dass sie keinem festgelegten Datenschema unterliegen.
- Web-Informationssysteme bestehen aus Datensätzen (auch Dokumente genannt), die eine Baumstruktur oder Graphstruktur haben.
- Um Daten aus Informationssystemen zu gewinnen, werden nicht nur Anfragesprachen, sondern auch Suchmaschinen eingesetzt.
- Datenströme stellen insbesondere im Web eine aufkommende Art von Informationssystemen dar.
- RDF, die Grundsprache des Semantic Web, stellt neue Ansprüche an Anfragesprachen, die bisher wenig erforscht worden sind.[1]

Die relationalen Anfragen aus der Datenbankforschung der 80-er Jahre des 20. Jahrhunderts eignen sich kaum zur Datenextraktion im Web, weil sie strikt eingehaltene Datenschemata voraussetzen, die es im Web nur sehr selten gibt, und weil sie die Baum- oder Graph-Struktur von Web-Dokumenten und -Metadaten nur indirekt und folglich umständlich wiedergeben können. Die Auswertung von relationalen Anfragen erfolgt nach Algorithmen, die im Gegensatz zu Suchverfahren kein Ranking der Antworten liefern. Relationale Anfragen sind auch für reaktive, Ereignis-basierte Systeme wenig geeignet, weil sie über keine Ausdrucksmittel für temporale Beziehungen verfügen, die zur Spezifikation von komplexen Ereignissen meist unabdingbar sind. Relationale Anfragen können auch nur schwer mit den Blank Nodes von RDF umgehen, die existenziell quantifizierten

François Bry
Institut für Informatik, Ludwig-Maximilians-Universität München, Oettingenstr. 67, 80538 München, Deutschland
Fon: +49 (0)89 21 80 93 10
Fax: +49 (0)89 21 80 93 11
E-Mail: bry@lmu.de

[1] Obwohl RDF und RDF-Anfragesprachen viel erforscht wurden.

Aussagen entsprechen. Mit relationalen Anfragen sind bestimmte Arten von Prädikat-Verkettungen, die in RDF/S-Anfragen oft natürlich sind, nicht so einfach ausdrückbar, wie es wünschenswert wäre.

Dieser Artikel berichtet über Forschungen zwischen 2002 und 2010 an der Universität München mit dem Ziel, Anfragesprachen an die oben genannten neuen Gegebenheiten des Web anzupassen. Er stellt die prototypischen Anfragesprachen Xcerpt, KWQL, XChange^{EQ}, RDFLog und RPL vor, die das deklarative logik-basierte Paradigma der relationalen Anfragen auf verschiedene Weisen an Web- und Ereignis-basierte Systeme anpassen:

- Die Web-Anfragesprache Xcerpt mit einer neuen Form von Unifikation für unvollständige Anfragen;
- Die Web-Anfragesprache KWQL für ein Spektrum von Anfragetypen, das von einfachen Schlüsselwort-Anfragen bis hin zu Strukturanfragen reicht, und die Web-Afragesprache KWRL zur Begründungsverwaltung,
- Die Ereignisanfragesprache XChange^{EQ} mit fünf Anfragedimensionen, darunter einer temporalen;
- Die regelbasierte Anfragesprache RDFLog zur uneingeschränkten Behandlung der unbekannten Daten, der sogenannten Blank Nodes, von RDF;
- Die Anfragesprache RPL für Pfadanfragen in RDF-Graphen.

Der Artikel ist wie folgt gegliedert. Nach dieser Einleitung werden im Abschnitt 2 Anforderungen an Web-Anfragesprachen eingeführt. Abschnitt 3 stellt eine erste Web-Anfragesprache namens Xcerpt vor, Abschnitt 4 zwei weitere namens KWQL und KWRL. Im Abschnitt 5 wird erklärt, wie zwei visuelle Anfragesprachen namens visXcerpt und visKWQL als graphische Wiedergabe der textuellen Anfragesprachen Xcerpt und KWQL realisiert wurden und warum dieser Ansatz zur Realisierung von visuellen Anfragesprachen vom Vorteil ist. Im Abschnitt 6 wird die Ereignis-Anfragesprache XChange^{EQ} vorgestellt. Im Abschnitt 7 wird eine erste Anfragesprache für RDF/S namens RDFLog eingeführt, die die Ableitung von Blank Nodes ermöglicht. Im Abschnitt 8 wird eine zweite Anfragesprache für RDF/S vorgestellt, mit der Eigenschafts-Verkettungen leicht ausdrückbar sind. In einem abschließenden Abschnitt 9 werden Forschungsperspektiven auf dem Gebiet der Web-Anfragesprachen vorgestellt.

Der Artikel fokussiert auf die Motivation sowie die Hauptmerkmale der erwähnten Sprachen. Für detaillierte Darstellungen der Anfragesprachen sowie deren Implementierungen und Eigenschaften wird auf bisherige Veröffentlichungen verwiesen.[2]

[2] In diesem Artikel wird aus Platz Gründen nur ein Teil der Literatur zu den behandelten Anfragesprachen zitiert. Für weitere Publikationen siehe: `http://www.pms.ifi.lmu.de/publikationen/`

2 Anforderungen an Web-Anfragesprachen

Im Folgenden werden Prinzipien eingeführt, die die Konzeption der in diesem Artikel vorgestellten Web-Anfragesprachen geleitet haben: Deklarativität, unvollständige Anfragen, unvollständige Antworten und Vielseitigkeit. Diese Prinzipien sind aus der Betrachtung der bisher vorgeschlagenen Web-Anfragesprachen [1,2] entstanden.

2.1 Deklarativität

Die Deklarativität ist sowohl für Web- als auch für Datenbankanfragesprachen von Vorteil. Was ist aber darunter zu verstehen? Und worin liegt der Vorteil für eine Anfragesprache, deklarativ zu sein? Deklarativ werden Programmier- und Anfragesprachen genannt, mit denen das Ergebnis einer Berechnung spezifiziert werden kann, ohne dabei die Berechnungsschritte festlegen zu müssen. Die Paradebeispiele für deklarative Anfragesprachen sind SQL, Datalog und Query-By-Example (kurz QBE).

SQL und Datalog können beide als logische Sprachen angesehen werden, obwohl SQL von der üblichen Syntax der Logik wesentlich abweicht: SQL ist eine Umsetzung des sogenannten Tupel-Kalküls, wogegen Datalog wie die klassische Logik auf einem sogenannten Bereichskalkül beruht. In einem Tupel-Kalkül stehen Variablen für Tupel, in einem Bereichskalkül für Tupel-Argumente. Mit SQL und Datalog werden Anfragen in zwei verschiedenen Weisen ausgedrückt, die beide als formale Umsetzung einer natürlichen Sprache angesehen werden können. Query-By-Example ist eine Anfragesprache, die aus der Gründerzeit der relationalen Datenbanken stammt und viel Ansehen erhielt und immer noch genießt. Query-By-Example wird jedoch in der Praxis nicht mehr verwendet. Mit Query-By-Example wird eine Anfrage durch das Ausfüllen eines oder mehrerer Relationsmuster ausgedrückt. Das Ausdrücken einer Anfrage in Query-By-Example erinnert also an das Ausfüllen eines Formulars: Das Formular gibt die Struktur der angefragten Datenbank wieder, das Ausfüllen die daraus zu extrahierenden Daten. Query-By-Example verfügt über logische Variablen, womit die selben Daten aus verschiedenen Relationen oder Tupeln verlangt werden können.

Logikbasierte Anfragen und Anfragen in Query-By-Example können als sehr ähnlich angesehen werden, wenn man vom eigentlichen Format der Grund-Anfragen absieht: Eine logikbasierte Anfrage ist textuell, eine Anfrage in Query-By-Example ist ein Diagramm. In Datalog sind die Grund-Anfragen textuelle atomare Formeln der Logik; in SQL sind sie eine textuelle Umsetzung der atomaren Formeln der Logik in einem Tupel-Kalkül; in Query-By-Example sind sie eine diagrammatische Umsetzung der atomaren Formeln der Logik.

Formal betrachtet, kann die Deklarativität einer Anfragesprache mit zwei Konzepten gleichgesetzt werden: Referenzielle Transparenz und Antwort-Abgeschlossenheit. Referenziell transparent heißt eine Programmier- oder Anfragesprache, wenn im selben Geltungsbereich von Variablen alle Vorkommen des gleichen Ausdrucks dieselbe Bedeutung haben. Antwort-abgeschlossen ist eine Anfragesprache, deren Antworten dieselbe Syntax wie variablenfreie Anfragen haben. Bekanntlich trägt die referenzielle Transparenz wesentlich zur Einfachheit der Programmierung bei. Die Antwort-Abgeschlossenheit trägt ebenfalls dazu bei, zum einen weil sie eine intuitive Formulierung von einfachen Anfragen ermöglicht, zum anderen weil sie eine einfache Verkettung oder Verschachtelung von Anfragen erlaubt.

SQL, Datalog und Query-By-Example sind referenziell transparent. Imperative Sprachen mit Zuweisungsoperation sind es nicht, ebenso wenig wie die verbreiteten Web-Anfragesprachen XPath und XQuery [1,3]. Datalog sowie die meisten Umsetzungen von Datalog, die für theoretische oder praktische Zwecke vorgeschlagen wurden, sind antwort-abgeschlossen. Prolog aber, das die Inspiration zu Datalog gewesen ist, ist wegen seiner Antwort-Substitutionen nicht antwort-abgeschlossen. SQL ist wegen seiner Syntaxunterschiede zwischen Anfragen und Antworten ebenfalls nicht antwort-abgeschlossen. Query-By-Example kann als antwort-abgeschlossen angesehen werden.

Ein erster Vorteil der Deklarativität einer Anfragesprache liegt darin, dass Anfragen in deklarativen Anfragesprachen leicht ausdrückbar sind. Der Vorteil ist von Belang, weil die meisten Informationssysteme auch von unerfahrenen Nutzern angefragt werden. Ein zweiter Vorteil besteht darin, dass die Deklarativität einer Anfragesprache eine Abstraktion von den Speicherstrukturen ermöglicht: werden die Daten in verschiedener Weise gespeichert, so müssen die Auswertungsverfahren an die jeweiligen Speicherstrukturen angepasst werden; die deklarativen Anfragen können aber gleich bleiben. Ein dritter Vorteil der Deklarativität einer Anfragesprache liegt darin, dass dieselbe Anfrage in verschiedener Weise ausgewertet werden kann, was Optimierungen wesentlich erleichtert.

Als vor ungefähr fünfzehn Jahren die ersten Web-Anfragesprachen XPath und XQuery konzipiert wurden, wurde auf eine Navigation durch baumartige Datensätze gesetzt. In der inhärenten Prozeduralität einer solchen Navigation liegt die Ursache dafür, dass XPath und XQuery nicht referenziell transparent sind. Die Entscheidung für das Navigations-Paradigma hatte jedoch einen guten Grund: Im Web werden meist unvollständige Anfragen benötigt, die weder in Datalog noch in SQL ausdrückbar sind.

2.2 Unvollständige Anfragen

Um mit SQL, Datalog oder Query-By-Example eine Anfrage auszudrücken, muss man die Struktur, das sogenannte Schema, der Datenbank kennen: ihre Relationen und die Argumente dieser Relationen. Es ist aber illusorisch, eine ähnliche Kenntnis für Web-Anfragen vorauszusetzen, weil es im Web kein verpflichtendes Schema gibt. Im Web kann jede Web-Seite jederzeit lokal strukturiert oder umstrukturiert werden. Web-Anfragesprachen müssen also das Ausformulieren von Anfragen auch dann ermöglichen, wenn die Struktur der angefragten Web-Dokumente nur teilweise bekannt ist. Man spricht von „unvollständigen Anfragen“. Es wird zum Beispiel nach Web-Seiten über Lehrveranstaltungen über „Künstliche Intelligenz“ gesucht, aus denen die Namen der Dozenten, die die Lehrveranstaltungen halten, und die Semester, wann die Lehrveranstaltung stattfindet, extrahiert werden sollen. Wie genau diese Web-Seiten strukturiert sind, soll in der Anfrage offen bleiben können, weil es meist weitgehend unbekannt ist.

2.3 Unvollständige Antworten

Werden relationale Datenbanken oder das Web angefragt, so sind häufig nicht alle Merkmale, die die gesuchten Datensätze spezifizieren, in der Antwort erwünscht. Sucht man zum Beispiel nach Flugverbindungen an einem bestimmten Datum, so wird häufig eine Antwort gewünscht, die dieses Datum nicht beinhaltet. Mit relationalen Datenbanken sagt man, dass die Argumente, die in der Antwort unerwünscht sind, „wegprojiziert“ werden sollen. Relationale Anfragesprachen bieten dafür einfache und intuitive Mitteln: SQL-Anfragen nennen explizit diejenigen Argumente der angefragten Relationen, die in der Antwort behalten werden sollen; Datalog- oder Prolog-Anfragen nennen explizit diejenigen Variablen, die für Relationsargumente stehen, die in der Antwort behalten werden sollen.

Konstrukte für unvollständige Antworten zu Web-Anfragen anzubieten, ist dagegen schwieriger, weil sowohl die angefragten Web-Dokumente als auch die durch eine Anfrage ausgewählten Daten als auch die Antworten eine Baum- oder Graph-Struktur haben statt der flachen Struktur relationaler Tupel.

2.4 Vielseitigkeit: Web und Semantic Web gleichartig anfragen können

Das Semantic Web, insbesondere mit RDF/S und der „Linked Open Data“-Initiative, verändert die Lage völlig, die zur Zeit der Entstehung der Web-Anfragesprachen XPath und XQuery geherrscht hat: jetzt müssen nicht nur Baumstrukturierte Dokumente angefragt werden, sondern auch

RDF/S-Graphen. Es ist fraglich, ob die Verwendung von unterschiedlichen Anfragesprachen, also von XPath und XQuery für HTML- und XML-Dokumente und von SPARQL für RDF/S-Dokumente, oder die Übersetzung von Web-Dokumenten in einen einzigen Formalismus, etwa von HTML- und XML-Dokumenten in RDF/S-Graphen, adäquat ist. Vielmehr erscheint es wünschenswert, erstens eine einzige Anfragesprache verwenden zu können, womit sowohl HTML- als auch XML- als auch RDF/S-Dokumente angefragt werden, zweitens in ein und derselben Anfrage mehrere Web-Dokumente von verschiedenen dieser drei Arten anfragen zu können. Solche Web-Anfragesprachen nennt man „vielseitig“ [4,3]. Der Einfachheit halber sollte eine vielseitige Web-Anfragesprache dasselbe Paradigma für die Anfrage von HTML-, XML- und RDF/S-Dokumenten umsetzen. Angesichts der signifikanten konzeptionellen Unterschiede zwischen HTML und XML einerseits und RDF/S andererseits, stellt die Vielseitigkeit einer Web-Anfragesprache kein einfaches Ziel dar.

2.5 Umstrukturierung, Gruppierung und Aggregierung

Ein typisches Beispiel für eine Umstrukturierung stellt die Anfrage an eine Liste von Veröffentlichungen dar, die nach Jahren gegliedert ist. Eine Anfrage kann alle oder nur einige Veröffentlichungen auswählen und das Ergebnis sortiert nach Autoren statt wie in der angefragten Liste nach Jahren verlangen. Können einige Veröffentlichungen mehrere Autoren haben, so ist die Umstrukturierung mit einer imperativen Sprache nicht einfach zu programmieren. Web-Anfragen ebenso wie relationale Anfragen verlangen oft, einige der ausgewählten Daten in der Antwort zu gruppieren oder zu aggregieren. Ein Beispiel für eine Gruppierung liefert eine Abwandlung des obigen Beispiels: Die Anfrage kann zudem verlangen, dass die Ergebnisse gruppiert nach Tagungen geliefert werden. Eine Aggregierung findet statt, wenn im obigen Beispiel anstatt der ausgewählten Veröffentlichungen ihre Gesamtanzahl oder die durchschnittliche Anzahl pro Autor oder pro Autor und Jahr angefordert wird.

Umstrukturierung, Gruppierung und Aggregierung sind insofern ähnlich, als sie alle auf einer Umgruppierung der von einer Anfrage ausgewählten Daten beruhen. Für Baum- oder Graph-strukturierte Datensätze, wie sie im Web und Semantic Web zu finden sind, ist diese Umgruppierung wesentlich schwieriger zu berechnen als für die flachen Tupel relationaler Datenbanken.

3 Xcerpt: Simulationsunifikation für unvollständige Anfragen

Xcerpt [5,6] ist eine praxisbezogene Anpassung von Datalog mit dem Ziel, die im vorigen Abschnitt genannten Anforderungen an Web-Anfragesprachen (Referenzielle Transparenz, Antwort-Abgeschlossenheit, unvollständige Anfragen, unvollständige Antworten, Vielseitigkeit, Umstrukturierung, Gruppierung und Aggregierung) zu erfüllen, ohne dabei die Deklarativität zu kompromittieren.

Xcerpt erweitert die atomaren Anfragen von Datalog in drei Weisen: nicht alle Argumente eines Tupels müssen in einer Anfrage explizit angegeben werden; die Reihenfolge der Argumente eines Tupels muss in einer Anfrage nicht eingehalten werden; eine Anfrage kann einen Teilterm anfordern, ohne dessen Schachtelungstiefe in den Daten spezifizieren zu müssen.

Die Xcerpt-Anfrage

```
Autor[ Vorname["Hans"], Nachname["Müller"] ]
```

steht zum Beispiel für XML-Dokumente der Gestalt:

```
<Autor>
   <Vorname>Hans</Vorname>
   <Nachname>Müller</Nachname>
</Autor>
```

Doppelte Klammerung [[ ]] drückt in Xcerpt unvollständige Argumentlisten aus. Die unvollständige Anfrage

```
Autor[[ Vorname["Hans"], Nachname["Müller"] ]]
```

liefert zum Beispiel als Antwort neben dem obigen XML-Dokument auch XML-Dokumente, die weitere Kind-Elemente enthalten, die in der Anfrage nicht genannt sind, wie etwa:

```
<Autor>
   <Vorname>Hans</Vorname>
   <Nachname>Müller</Nachname>
   <Email>hans.mueller@xfp.net</Email>
</Autor>
```

Wird die Klammerung [ ] bzw. [[ ]] in { } bzw. {{ }} geändert, so ist die Reihenfolge der angegebenen Kind-Elemente (oder Argumente) irrelevant. Die Anfrage

```
Autor{ Nachname["Müller"], Vorname["Hans"] }
```

kann somit das erste der obigen XML-Dokumente als Antwort liefern, die Anfrage

```
Autor{{ Nachname["Müller"], Vorname["Hans"] }}
```

kann beide obigen XML-Dokumente liefern.

Mit dem Sprachkonstrukt `desc`, für „descendant“, muss das entsprechende Element kein Kind sein. Es darf in beliebiger Tiefe in der Antwort vorkommen. Die Anfrage `a[desc b]` liefert also als Antworten alle Dokumente mit Wurzel-Element `a`, die in beliebiger Schachtelungstiefe ein `b`-Element enthalten, etwa die folgenden XML-Dokumente:

```
<a>
  <b />
<a>
```

```
<a>
  <c>
    <b />
  </c>
<a>
```

Die verschiedenen oben genannten Sprachkonstrukte von Xcerpt sowie einige weitere, die hier nicht eingeführt werden, können weitgehend beliebig kombiniert werden. Besonderheiten von XML wie etwa Verweise innerhalb eines

Dokuments durch Hyperlinks und ID-IDREF-Attribute werden von Xcerpt berücksichtigt. Die Anfragesprache Xcerpt verfügt wie Datalog über logische Variablen, wobei eine Variable innerhalb einer atomaren Anfrage an einen Teilterm gebunden werden kann wie etwa die Variable `Aut` in der Anfrage:

```
var Aut as
   Autor[
      Vorname[var Vor],
      Nachname[var Nach]
   ]
```

Zur Auswertung der unvollständigen Anfragen von Xcerpt wurde die klassische Unifikation durch eine neue, sogenannte Simulationsunifikation [7] ersetzt. Die Simulationsunifikation lässt zyklische Terme zu, wie sie in XML-Dokumenten unter Verwendung von Hyperlinks oder ID-IDREF-Attributen möglich sind, und verwendet Memoisierungstechniken für die Auswertung.

Die Simulationsunifikation über uneingeschränkten Termen ist NP-vollständig. Das ist überraschend angesichts der Ähnlichkeit von Simulationsunifikation ungeordneter Terme mit Tiefenunvollständigkeit (durch `desc` ausgedruckt) zur Inklusion ungeordneter Bäumen (oder „minor containment“ auf Bäumen), die eine NP-vollständige [8] Verallgemeinerung von Baum- und Teilbaum-Isomorphie, die Kanten-Verschmelzung zulässt. Werden die Terme, die unifiziert werden sollen, eingeschränkt, so ergeben sich bessere Komplexitätsklassen für die Simulationsunifikation: sie ist zum Beispiel polynomiell über geordneten oder ungeordneten Baumstrukturierten Termen. Präzisere und weitere Komplexitätsergebnisse für die Simulationsunifikation mit Termen aus verschiedenen nützlichen Anfrageklassen finden sich in [9].

Auf der Grundlage der Simulationsunifikation wurde eine Subsumption [10] definiert, die entscheidbar ist.

Eine Xcerpt-Anfrage trennt strikt zwischen Daten-Auswahl und Umstrukturierung der Antwort, die auch Antwort-Konstruktion genannt wird. Eine Xcerpt-Anfrage hat eine Regel-Gestalt, wobei der Rumpf (die Prämisse) der Regel die Daten-Auswahl ausdrückt, während der Kopf (die Konklusion) die Antwort-Konstruktion spezifiziert. Diese strikte Trennung, die mit Prolog nicht immer möglich ist und die XQuery fremd ist, erleichtert die Verwendung der Anfragesprache.

Die Semantik von Xcerpt ist in der traditionellen Weise der Logikprogrammierung durch eine Minimalmodell-Theorie definiert. Eine Abweichung von der üblichen Minimalmodell-Theorie liegt dabei jedoch darin, dass die Minimalmodell-Theorie von Xcerpt nur Atome kennt. Die Semantik von Xcerpt unterscheidet also nicht zwischen Relations- und Funktionssymbolen: In Xcerpt sind alle Terme Atome. Folglich kann Xcerpt wegen der geschachtelten Terme syntaktisch als höherstufige Logiksprache angesehen werden. Die Semantik von Xcerpt ist aber erststufig.

Xcerpt verfügt über fortgeschrittene Sprachkonstrukte, die die Formulierung von komplexen Anfragen, insbesondere mit unvollständigen Antworten, vereinfacht. Zur Gruppierung und Aggregierung können in Prolog vordefinierte Prädikate wie „setof“ und „bagof“ verwendet werden, die aber nur im Rumpf von Prolog-Regeln erlaubt sind. Xcerpt bietet zu diesem Zweck vergleichbare Sprachkonstrukte an, erlaubt diese aber ausschließlich in der Antwort-Konstruktion, also im Kopf von Regeln. Die Verwendung von Sprachkonstrukten zur Gruppierung und Aggregierung in der Antwort-Konstruktion trägt insofern zur Deklarativität der Anfragesprache bei, dass die Daten-Auswahl lediglich eine Antwortmenge spezifiziert, jedoch nicht, welcher Teil dieser Menge in die Antwort übernommen und gegebenenfalls umstrukturiert oder aggregiert werden soll [5].

Für Xcerpt ist ein anspruchsvolles Laufzeitsystem [11] konzipiert worden, welches besondere Datenstrukturen und Algorithmen zur Datenextraktion aus Baum- und Graph-strukturierten Datensätzen, wie XML sie zulässt, verwendet.

Weitere Forschungen um Xcerpt waren einem Typsystem gewidmet [12].

Xcerpt erfüllt die Anforderungen vom Abschnitt 2, ist aber eine wissenschaftliche Untersuchung geblieben. Ein Grund dafür ist, dass Anfragesprachen wie Xcerpt (oder wie XQuery) keine einfache Suche wie mit einer Suchmaschine unterstützen, die aber für Web-Anfragen oft erwünscht ist. So wie ein einfaches „Hello World“-Programm in Java einiges mehr an Java-Kenntnissen abverlangt als nur die Angabe dieses Strings (zumindest wenn man das Programm ganz verstehen will), so kann man mit Xcerpt (wie auch mit XQuery) nicht einfach nur einen String wie „Künstliche Intelligenz“ angeben, um nach Web-Seiten zu suchen, die diesen String enthalten. Genau dieser Mangel war der Anlass für die Entwicklung der Web-Anfragesprache KWQL, die im nächsten Abschnitt vorgestellt wird.

4 KWQL und KWRL: Von Schlüsselwort- zu Strukturanfragen und Begründungsverwaltung

Die Anfragesprache KWQL [13, 14] wurde mit dem Ziel entwickelt, sowohl einfache Schlüsselwort-Anfragen, wie sie durch Suchmaschinen verbreitet sind, als auch komplexere Strukturanfragen zu ermöglichen. Zudem wurde im KWQL-Projekt untersucht, wie ein Ranking à la PageRank auf die Dokument-Struktur, auf die Hypertext-Struktur einer Dokumentensammlung und auf semantische Beziehungen innerhalb eines Dokuments und zwischen Dokumenten erweitert werden kann [15].

KWQL ermöglicht einfache Schlüsselwort-Anfragen, zum Beispiel für das Finden von Web-Dokumenten, die den String „Künstliche Intelligenz“ enthalten: ein solcher String *ist* bereits eine KWQL-Anfrage. KWQL bietet zudem sogenannte Label-Schlüsselwort-Anfragen wie etwa

`autor: "Müller"`, die durch benutzerdefinierte und eine kleine Auswahl an vordefinierten Labels sowohl Struktur- als auch Semantik-Anfragen ermöglichen. Die Formulierung solcher Anfragen in KWQL ist extrem einfach, ihre Schwierigkeit wächst erst mit der Komplexität der Struktur oder der semantischen Beziehungen der Dokumente, die als Antwort geliefert werden sollen.

KWQL wurde für ein Semantic Wiki entwickelt, welches sowohl eine feste, im Voraus bekannte, Auswahl an semantischen Beziehungen anbietet als auch seinen Nutzern die Möglichkeit gibt, weitere semantische Beziehungen zwischen Wiki-Seiten sowie Text-Fragmenten zu definieren. Eine semantische Beziehung, etwa die Autorenschaft einer Web-Seite, wird in KWQL genauso durch ein Label ausgedrückt wie eine Struktur-Beziehung, etwa eine Eltern-Kind- oder Vorfahre-Nachfahre-Beziehung zwischen Struktur-Elementen. Die analoge Behandlung von semantischen und Struktur-Beziehungen macht die Anfragesprache KWQL einfach und anschaulich für ihre Nutzer. Sie ermöglicht zudem eine weitgehend ähnliche Behandlung beider Arten von Beziehungen während der Anfrageauswertung.

Der Vorteil einer Anfragesprache des Label-Schlüsselwort-Ansatzes gegenüber einer logischen Schreibweise à la Datalog, Prolog oder Xcerpt liegt darin, dass keine Struktur explizit angegeben werden muss, wo keine zum Ausdruck der Anfrage nötig ist. Dies mag zunächst als wesentliche Abweichung von einer logischen Sprache erscheinen, stellt aber einen Eingriff nur in die Syntax der Sprache dar. Die Semantik der Sprache kann ohne weiteres ähnlich spezifiziert werden, wie die einer Sprache mit traditioneller „logischer“ Syntax.

Eine Erweiterung von PageRank namens PEST, ein Kürzel für „term-Propagation using Eigenvector computation over wiki-STructures“ wurde entwickelt [14, 15], das sowohl semantische als auch strukturelle Beziehungen innerhalb von und zwischen Dokumenten berücksichtigt. Ein Dokument mag zum Beispiel durch eine semantische Beziehung mit einem zweiten verbunden sein, welches seinen Autor beschreibt, außerdem durch eine strukturelle Beziehung mit einem dritten Dokument, in das es als Teildokument eingebettet ist. Die Grundidee von PEST ist, neben der Hyperlinkstruktur des Dokumentenbestands weitere semantische oder strukturelle Beziehungen in ähnlicher Weise zu berücksichtigen und durch eine Gewichtung in ein einziges Ranking zu überführen. Im Gegenteil zu PageRank, welches PEST erweitert, ist das PEST-Ranking insofern abhängig vom Suchbegriff, dass eine semantische Struktur im Gegenteil zur Hypertext-Struktur abhängig vom Suchbegriff sein kann. Für jeden semantischen Kontext eines Suchbegriffes muss mit PEST ein Eigenvektor berechnet werden. Sind nur wenige semantische Kontexte ausreichend, so genießt PEST den selben Vorteil wie PageRank, nämlich das Ranking im Voraus und unabhängig vom Suchbegriff berechnen zu können. Sind aber sehr viele semantische Kontexte nötig oder sind gemeinsame semantische Kontexte für mehrere Suchbegriffe ausgeschlossen, so kann das PEST-Ranking im Gegenteil zum PageRank nicht im Voraus berechnet werden. Dieser Nachteil ist für alle bisher bekannte semantische Rankings kennzeichnend.

Wie erfüllt KWQL die Anforderungen an einer Web-Anfragesprache vom Abschnitt 2? KWQL ist referentiell transparent aber nicht wirklich antwort-abgeschlossen, weil eine Label-Schlüsselwort-Anfrage von der eigentlichen Struktur von Web-Seiten wesentlich abstrahiert. Unvollständige Anfragen und Antworten sind in KWQL möglich. KWQL ist insofern vielseitig, dass sowohl Wiki-Seiten wie RDF-/S-Daten über diese Seiten mit KWQL angefragt werden können. KWQL wurde aber nicht für eine direkte Anfrage von RDF/S-Daten, die keine Web-Seite beschreiben, konzipiert. Umstrukturierungen, Gruppierungen und Aggregation sind in KWQL möglich.

KWRL [16] ist eine Regelsprache, die mit dem Ziel entwickelt wurde, die Anforderungen eines Semantic Wiki an eine Regelsprache zu erfüllen. KWRL beruht auf stratifiziertem Datalog$^{\neg}$, wird mit einem originellen vorwärtsschließenden Verfahren ausgewertet, materialisiert die abgeleiteten Fakten und aktualisiert bei Änderungen von Fakten oder Regeln dementsprechend die materialisierten Fakten. Die praktische Umsetzung von KWRL verwendet an Stelle der konjunktiven Anfragen von Datalog die Anfragen von KWQL – siehe Abschnitt 4.

Eine Kernfunktionalität eines Semantic Wiki ist die Versionsverwaltung. Eine Regelsprache für ein Wiki muss also in der Lage sein, Schlussfolgerungen entsprechend eines früheren Standes des Wiki zu berechnen. Dafür sind in den 80er Jahren des 20. Jahrhunderts sogenannte Methoden zur Begründungsverwaltung entwickelt worden. Die bisherigen Methoden der Begründungsverwaltung haben jedoch eine Eigenschaft, die im Kontext eines Wiki schwer annehmbar ist: Sie verwenden auf eine Überstzung der von den Nutzern definierten Regeln in ein Ziel-Regelprogram. Eine solche Übersetzung ist für ein Wiki nachteilhaft, weil die Regeln häufig aktualisiert werden können, was zu zeitaufwendigen Übersetzungen führen kann.

Um diesen Nachteil zu beheben, sind im KWRL-Projekt ein paar vorwärtsschließende Verfahren entwickelt worden, die in verschiedenen Weisen eine Begründungsverwaltung ohne Übersetzung des Regelprograms ermöglichen [16, 17]. Dafür wurde der bekannte Fixpunkt-Operator der Logik-Programmierung, welche für Fakten-Mengen definiert ist, auf Fakten-*Multi*mengen angepasst [17].

KWRL lässt zudem Regeln zu, die als Köpfe (oder Konklusionen) die nicht erfüllbare Formel $\bot$ haben. Eine solche Regel drückt die Negation ihres Rumpfs (oder ihrer Prämisse) aus und wird deswegen ”Denial”genannt. Denials ermöglichen, Integritätsbedingungen, wie sie aus Datenban-

ken bekannt sind, auszudrücken, womit die Regelsprache die Bedürfnisse von Wiki-Anwendungen erfüllt. Mit Denials kann ein Regelprogramm widersprüchlich sein. Im Kontext eines Wiki wird ein widersprüchliches Regelprogram als natürlicher Schritt in seiner Entwicklung angesehen. Folglich wurde ein einfacher und intuitiver konstruktiver Ansatz zur Parakonsistenz für KWRL entwickelt. KWRL stellt folglich eine Logik dar, die obwohl ähnlich ausdruckskräftig wie die Prädikatenlogik erster Stufe keine Widerspuchsbeweise zulässt, was zur leichteren Verständnis der KWRL-Regelprogramme beiträgt.

Die Sprache KWRL ist wie Datalog referentiell transparent und antwort-abgeschlossen. Wie Datalog ermöglicht KWRL, unvollständige Anfrage und Antworte auszudrücken. KWRL ist insofern vielseitig, dass KWRL Anfragen an die RDF/S-Metadaten eines Wiki zulässt. KWRL deckt aber nur einen Teil von RDF/S und berücksichtigt RDF/S-Konstrukte wie Blank Nodes nicht. Wobei KWRL wie Datalog eine Umstrukturierung der Antworten ermöglicht, fehlt KWRL die nötige Sprachkonstrukte zum Ausdruck von Gruppierungen und Aggregierungen. KWRL um solche Sprachkonstrukte zu erweitern, wäre jedoch keine große Herausforderung.

5 Visuelle Anfragesprachen als Wiedergabe textueller Anfragesprachen

Die textuellen Anfragesprachen Xcerpt und KWQL wurden als Grundlage für visuelle Anfragesprachen namens visXcerpt und visKWQL eines neuen Typs verwendet. Statt neue visuelle Sprachen zu entwickeln, wurde die Deklarativität von Xcerpt und KWQL genutzt, um die visuellen Anfragesprachen visXcerpt und visKWQL als diagrammatische zweidimensionale Wiedergabe der textuellen Anfragesprachen Xcerpt und KWQL zu realisieren. Der Artikel [18] zeigt, wie visXcerpt mit einer geringfügigen Erweiterung der Formatierungssprache Cascading Style Sheets (CSS) für XML und HTML implementiert werden konnte.

Der Ansatz, eine visuelle Anfragesprachen als Wiedergabe, also praktisch nur als alternative Syntax, einer textuellen Anfragesprache zu realisieren, ist in erster Linie wegen der Einfachheit der Implementierung der visuellen Sprache vorteilhaft. Andererseits hat er zur Folge, dass visXcerpt und visKWQL Symbole verwenden, die sich von den Symbolen einer textuellen Anfragesprache nicht wesentlich unterscheiden, was Liebhaber von visuellen Sprachen als Nachteil ansehen könnten. Sie könnten dem Ansatz vorwerfen, nicht mehr als eine zweidimensionale textuelle Sprache ähnlich wie Freges *Begriffsschrift* zu sein.

Diesem Einwand stehen aber zwei weitere sehr wesentliche Vorteile des Ansatzes gegenüber. Erstens liegen der visuellen und der textuellen Sprache dasselbe Programmierparadigma zugrunde, was das Erlernen beider Sprachen erleichtert. Zweitens erlaubt der Ansatz „Rundfahrten“ während der Entwicklung von Anfragen, das heißt, beliebige Wechsel zwischen visueller und textueller Darstellungsweise, was sich als äußerst hilfreich erwiesen hat [14].

visXcerpt und vbisKWQL erfüllen die Anforderungen an Web-Anfragesprachen vom Abschnitt 2 genauso wie die textuellen Anfragesprachen Xcerpt beziehungsweise KWQL, von denen sie abgeleitet worden sind.

6 XChange^{EQ}: Die zeitliche und weitere Dimensionen von Ereignisanfragen

Mit der Ereignis-Anfragesprache XChange^{EQ} wurde untersucht, wie sich die Web-Anfragesprache Xcerpt auf das sogenannte Complex Event Processing (kurz CEP) anpassen lässt. Unter Complex Event Processing versteht man die Erkennung von komplexen Mustern in einem Ereignisstrom, welcher möglicherweise zu groß ist, um vor der Anfrageauswertung gespeichert zu werden. Die Beschreibung von komplexen Ereignissen entspricht Anfragen an den Ereignisstrom. Die Erkennung komplexer Ereignisse in einem Ereignisstrom entspricht der Anfrageauswertung mit zwei Besonderheiten: zum einen weil Ereignisse zeitabhängig sind, zum anderen weil die Anfrageauswertung aus Effizienzgründen inkrementell sein muss.

Die Zeitabhängigkeit der Ereignisse ergibt sich aus den Anwendungen, wofür CEP nötig ist: die Sensor-generierten atomaren Ereignisse, aus denen sich eine Feuergefahr in einer U-Bahn-Station oder eine Tsunami-Gefahr an einer Küste oder andere komplexe Ereignisse erkennen lassen, sind eben datiert. Die Auswertung zur Erkennung von komplexen Ereignissen muss inkrementell geschehen, damit sie stufenweise beim Ankommen der neuesten atomaren Ereignisse stattfindet.

Die meisten Anfragesprachen, die bisher für das CEP vorgeschlagen wurden, beruhen auf Operatoren, die eine sowohl boolesche als auch zeitliche Bedeutung haben [19], wie etwa ein „und-dann“-Operator zur Konjunktion und Sequenzierung. Solche Operatoren haben den Vorteil, den Ausdruck von einfachen Ereignissen zu erleichtern. Sie haben jedoch den Nachteil, den Ausdruck von komplexen Ereignissen wesentlich zu erschweren.

Auch die CEP-Anfragesprache XChange [20], die eine Anpassung von Xcerpt an das CEP ist, beruht auf derartigen kombinierten boolesch-zeitlichen Operatoren. Aber ihre Weiterentwicklung XChange^{EQ} [21] verzichtet aus den obigen Gründen gänzlich darauf. Statt dessen ordnet sie Sprachkonstrukte für boolesche und für zeitliche Beziehungen zwei strikt voneinander getrennten sogenannten „Dimensionen“ zu. Weitere solche „Dimensionen“ für Ereignisanfragen wurden durch eine Untersuchung von Anwendungsfällen identifiziert, insgesamt die folgenden fünf:

- Daten-Extraktion wie etwa durch die Variablen einer atomaren Anfrage in Datalog oder Prolog;
- Boolesche Komposition von Ereignissen wie „und“ oder „nicht“;
- Temporale Komposition von Ereignissen wie „vor“ oder „während“;
- Ereignis-Akkumulation um Daten aus dem Ereignisstrom über einen Zeitraum zu sammeln;
- Ursprung (zum Ausdruck der Kausalität) von Ereignissen.

Die Semantik von XChange^{EQ} ist im Stil der Minimalmodell-Theorie und Fixpunkttheorie der Logikprogrammierung definiert [21,22]. Eine wesentliche Erweiterung war nötig, um die temporale Dimension von Ereignissen zu berücksichtigen.

XChange^{EQ} erfüllt die Anforderungen an Web-Anfragesprachen vom Abschnitt 2 genauso wie Xcerpt.

7 RDFLog: Existenzielle Ableitungen

Die Sprache RDF/S zur Modellierung von Meta-Daten im Web bietet sogenannte *Blank Nodes* an. Ein Blank Node steht für etwas, was existiert und Eigenschaften besitzen mag, aber nicht ausreichend bekannt ist, um über eine Identität zu verfügen. Die Semantik von RDF/S legt zudem fest, dass zwei Blank Nodes, die durch ihre Eigenschaften nicht unterschieden werden können, nicht als verschiedene Entitäten behandelt werden sollen. Blank Nodes können folglich als existenzielle Variablen angesehen werden.

Entwickelt man eine Regelsprache für RDF/S, so bietet sich an, Blank Nodes nicht nur in Regelrümpfen (Regelprämissen) zuzulassen, sondern auch in Regelköpfen (Regelkonklusionen). So kann eine Regel die bedingte Existenz eines unbestimmten, nicht näher bekannten Objekts ausdrücken. Es liegt nahe, in einer solchen Regelsprache Blank Nodes abhängig von universell quantifizierten Variablen spezifizierbar zu machen. So kann zum Beispiel ausgedrückt werden, dass jede Lehrveranstaltung von einem (unbekannten) Dozent gehalten wird. Logisch entspricht dies einer Folge von All- und Existenzquantoren. Diese sollten möglichst in beliebiger Reihenfolge aufeinander folgen können, um die Ausdruckskraft der Regelsprache nicht unnötig einzuschränken. Andernfalls wäre nicht gesichert, ob Regeln wie die folgende ausdrückbar sind: *„Zu jeder Vorlesung (*lecture*) gibt es eine Übung (*course*), die von jedem Student besucht wird, der die Vorlesung besucht.“*

RDFLog [9] ist eine RDF/S-Regelsprache, die eine uneingeschränkte Quantorenreihenfolge zulässt, das heißt, uneingeschränkte Abhängigkeiten der Blank Nodes von universellen Variablen ermöglicht. Das obige Beispiel kann zum Beispiel wie folgt in RDFLog ausgedrückt werden:

$$\forall\ lec\ \exists\ crs\ \forall\ stu$$

$$(lec \text{ rdf:type uni:lecture}) \wedge (stu \text{ uni:attends } lec) \Rightarrow$$
$$(crs \text{ uni:practices } lec) \wedge (stu \text{ uni:attends } crs)$$

RDFLog ist in dieser Hinsicht ausdrucksstärker als die RDF/S-Anfragesprache SPARQL, eine *W3C recommendation* zur Anpassung von SQL an RDF/S. Mit SPARQL kann man zwar Blank Nodes in einer Anfrage ausdrücken, aber keine, die wie im obigen Beispiel von universellen Variablen abhängen. Mit SPARQL kann man auch keine Regel ausdrücken.

Die deklarative Semantik von RDFLog [9] baut auf der Semantik von RDF/S auf: Sie wird unter Verwendung der Schlussfolgerungsbeziehung (*entailment*) von RDF/S definiert. Diese Abweichung vom üblichen Ansatz der Logikprogrammierung, der Minimalmodell-Semantik, ist durch eine syntaktischen Idiosynkrasie von RDF/S begründet: einerseits verbietet RDF/S gewisse Tripel mit Literalen als Tripel-Subjekt oder Blank Nodes als Tripel-Prädikat; andererseits hätten syntaktische Einschränkungen der Regel-Sprache, um die Ableitung solcher verbotenen Tripel zu verhindern, allzu weitgehende Folgen auf die Ausdruckskraft der Sprache.

Die operationale Semantik von RDFLog [9] behandelt Blank Nodes durch Skolemisierung, damit eine vorwärtsschließende Fixpunkt-Berechnung oder eine rückwärtsschließende Resolutionsmethode verwendet werden kann. Durch eine sogenannte Ent-Skolemisierung werden anschließend die generierten Skolem-Terme in Blank Nodes zurückgewandelt, um der RDF/S-Syntax zu genügen. Während des Schließens können in Zwischenergebnissen Literale als Tripel-Subjekte und Blank Nodes als Tripel-Prädikate vorkommen. Die letztendlich abgeleiteten Tripel halten sich jedoch an die RDF/S-Syntax, die Tripel mit solchen Komponenten verbietet. Eine Implementierung hat gezeigt [9], dass die operationale Semantik von RDFLog zu durchaus annehmbaren Auswertungszeiten führt.

RDFLog ist eine Erweiterung von Datalog. Folglich ist RDFLog wie Datalog deklarativ – referentiell transparent und antwort-abgeschlossen – und ermöglicht, unvollständige Antworten und Umstrukturierungen auszudrücken. Wie Datalog kann RDFLog weder unvollständigen Anfragen, noch Grupierung, noch Aggregierung ausdrücken und ist nicht vielseitig. Die Behandlung von Blank Nodes, die mit RDFLog eingeführt wurde, liesse sich ohne Schwierigkeiten auf eine Anfragesprache wie Xcerpt übertragen, die die Anforderungen vom Abschnitt 2 erfüllt.

8 RPL: Uneingeschränkte Pfade durch RDF/S-Graphen

Einerseits sind RDF/S und Datalog sehr ähnlich, weil beide Sprachen auf Tupeln beruhen. Andererseits sind bestimmte Arten von Verkettungen, die in RDF/S-Anwendungen natürlich sind, in Datalog nicht so einfach ausdrückbar wie

es wünschenswert wäre. Enthält zum Beispiel ein RDF/S-Graph zur Darstellung eines digitalen Netzwerks ein Prädikat „kennt“, so sind Anfragen über indirekte, in der Länge begrenzte oder unbegrenzte „kennt“-Beziehungen natürlich. Das Ausdrucksmittel für solche indirekten Beziehungen in Regelsprachen ist die Rekursion. Aber ungeübte Nutzer empfinden Rekursion häufig als „schwierig“, als schwer zu verstehen oder gar zu formulieren.

Einige Anfragesprachen, die für RDF/S vorgeschlagen worden sind, haben das Problem erkannt und Lösungen dafür angeboten [1]. Die meistverbreitete Lösung liegt darin, die Anfragesprache um einen Kleene-Stern-Operator und weitere ähnliche Operatoren für begrenzte oder unbegrenzte Prädikat-Verkettungen zu erweitern. Die RDF/S-Anfragesprache SPARQL verfügt weder über solche Operatoren noch über Rekursion und kann deshalb Prädikat-Verkettungen beliebiger Länge nicht ausdrücken.

RPL [23] ist eine Anfragesprache für RDF/S mit Verkettungsoperatoren, die Prädikat-Verkettungen einfach ausdrückbar machen. RPL kann als Anpassung sowohl von konjunktiven Datalog-Anfragen als auch von XPath-Ausdrücken angesehen werden. Die Verkettungsoperatoren von RPL entsprechen einer eingeschränkten Rekursion, deren Auswertung kaum mehr Zeit als die Auswertung von konjunktiven Anfragen verlangt. RPL ist die bisher ausdrucksstärkste RDF/S-Regelsprache, deren Auswertung in polynomieller kombinierter Komplexität möglich ist [23].

Wie erfüllt RDL die Anforderungen vom Abschnitt 2? RPL ist referentiell transparent und bis auf die Darstellung von Tupelmengen antwort-abgeschlossen. Unvollständige RDF/S-Anfragen und -Antworten können in RPL ausgedruckt werden. RPL ist nur für RDF/S konzipiert worden und somit nicht vielseitig. Umstruktierungen lassen sich in RPL spezifizieren, jedoch in Ermangelung der dafür nötigen Sprachkonstrukten weder Gruppierung noch Aggregation. Die nötige Erweiterungen dafür wären allerdings unproblematisch.

9 Forschungsperspektiven

Das bunte Gemisch an Web-Anfragesprachen in den vorangehenden Abschnitten wirft die Frage auf, wohin diese Forschung führen soll. Deshalb sei zunächst an ihren Hintergrund erinnert. Wie die meisten Fachgemeinschaften der Informatik, haben die Datenbankler und Entwickler von Anfragesprachen am Anfang der 90-er Jahre des 20. Jahrhunderts die Bedeutung des Web für ihre Fachgebiete völlig unterschätzt. Gegen Ende der 90-er Jahre haben sie dann die relativ schnelle Entwicklung von XQuery eingeleitet. Die Erwartung war, mit XQuery zum einen das Web einigermaßen in den Schoß der Datenbanksysteme zurück zu führen, zum anderen einen ähnlichen Erfolg wie mit SQL einzuleiten.

Die in diesem Artikel vorgestellten Web-Anfragesprachen sind aus der Überzeugung entstanden, dass XQuery nicht die ultimative Web-Anfragesprache sein kann. Das Ziel der in diesem Artikel vorgestellten Forschung war, mit Sprachansätzen und -konzepten zu experimentieren, die über XQuery hinausgehen. Zur Forschungsperspektive stellen sich damit die Fragen, ob nun eine oder mehrere, wenn nicht ultimative, so doch zumindest bessere Web-Anfragesprachen konzipiert werden könnten und ob weitere solche Experimente nötig sind.

Bessere und vielseitige, das heißt XML- und RDF/S-fähige, Web-Anfragesprachen sind sicherlich deswegen heute vorstellbar, weil das logische Programmierparadigma von SQL und Datalog für Web-Anfragen nicht nur möglich, sondern auch gegenüber XQuery vorteilhaft ist, wie unter anderem die Untersuchungen belegen, die in diesem Artikel vorgestellt wurden. Abweichungen von der Standardunifikation wie die Simulationsunifikation und die „rich unification“ genannten Ansätze [24] erscheinen für Web-Anfragesprachen vielversprechend. Eine „Unifikation modulo einer Ontologie“ würde vermutlich in einer vielseitigen Anfragesprache eine sehr nützliche Verbindung zwischen Objekt- und Meta-Daten ermöglichen. Offene Fragen dabei sind, welche Einschränkungen der Ontologie-Sprache eine entscheidbare und möglichst in nahe linearer Zeit berechenbare Unifikation ermöglichen würden, ohne dabei die Ausdruckskraft der Ontologie-Sprache allzu sehr einzuschränken.

Der Autor hält es also durchaus für möglich, heute bessere Web-Anfragesprachen als XQuery zu konzipieren. Darüber hinaus meint er aber, dass weitere Experimente nötig sind, weil die Zukunft von Web-Anfragesprachen unter anderem in einer ortsverteilten Auswertung und im Datenparallelismus liegt. Dazu müssen unter anderem noch folgende Fragen beantwortet werden: Wie sollen Regelprogramme in einem Rechnernetz statisch (das heißt, zur Übersetzungszeit) sowie dynamisch (das heißt, während der Auswertung) verteilt werden? Wie kann der Ansatz der Daten-Parallelisierung für das Vorwärts- und Rückwärtsschließen sowie für Unifikationsalgorithmen verwendet werden?

Unabhängig von solchen konzeptionellen Gesichtspunkten wäre es wünschenswert, Web-Anfragesprachen nicht lediglich als Proof of Concept-Prototypen zu entwickeln, sondern auch als Open Source-Software nachhaltig einer breiten Öffentlichkeit anzubieten und dadurch ihre Erprobung in mehreren Anwendungsfällen zu ermöglichen. Die offenkundige Diskrepanz zwischen den dafür notwendigen personellen Resourcen und den Gegebenheiten in europäischen Universitäts-Umgebungen eröffnet eine weitere Dimension von Fragen der Forschungsperspektiven.

Danksagung Der Autor bedankt sich bei Norbert Eisinger, Tim Furche und bei den Gutachtern für ihre Hilfe bei der Fertigstellung dieses Artikels. Er bedankt sich bei den folgenden Personen, die an der in

diesem Artikel vorgestellten Forschung mitgewirkt haben: Liviu Badea, James Bailey, Sacha Berger, Oliver Bolzer, Simon Brodt, Emmanuel Coquery, Włodzimierz Drabent, Michael Eckert, Norbert Eisinger, Tim Furche, Georg Gottlob, Andreas Hartl, Steffen Hausmann, Christoph Koch, Jakub Kotowski, Paula Kröner (ehemals Pătrânjan), Clemens Ley, Benedikt Linse, Jan Małuszyński, Massimo Marchiori, Bruno Marnette, Dimitris Plexousakis, Olga Poppe (ehemals Yestekhina), Sebastian Schaffert, Andreas Schröder, Klara Weiand, Antonius Weinzierl, Christoph Wieser, Artur Wilk und Harald Zauner.

Die Durchführung dieser Forschung wurde unterstützt durch Fördermittel der Europäischen Union im Rahmen des Exzellenznetzwerkes REWERSE (Referenz 506779 im 6. Rahmenprogramm), des Integrierten Projektes KiWi (Referenz 211932 im 7. Rahmenprogramm) und des integrierten Projektes EMILI (Referenz 242438 im 7. Rahmenprogramm) sowie der Deutschen Forschungsgemeinschaft (DFG).

François Bry hat in Paris studiert und wurde 1981 promoviert. 1981–1982 nahm er an der Entwicklung eines Textverarbeitungssystems teil. 1983–1984 untersuchte er statistische Datenbanken. 1985 wechselte er nach München zum ECRC, wo er bis 1993 über deduktive Datenbanken, Logikprogrammierung und automatische Deduktion forschte. 1994 wurde er zum Professor der Universität München berufen. Er forscht über Anfragesprachen, semantische Suche und soziale Medien.

Literatur

1. James Bailey, François Bry, Tim Furche, and Sebastian Schaffert. Web and Semantic Web Query Languages: A Survey. In *Reasoning Web, First International Summer School*, volume 3564 of *LNCS*. Springer-Verlag, 2005.
2. Tim Furche, Benedikt Linse, François Bry, Dimitris Plexousakis, and Georg Gottlob. RDF Querying: Language Constructs and Evaluation Methods Compared. In *Reasoning Web, Second International Summer School 2006*, volume 4126 of *LNCS*. Springer-Verlag, 2006.
3. François Bry, Tim Furche, and Klara Weiand. Web Queries: From a Web of Data to a Semantic Web. In *Proceedings of 10th International Conference on Web Systems Engineering (WISE)*, volume 5802 of *LNCS*. Springer-Verlag, 2009.
4. François Bry, Tim Furche, Liviu Badea, Christoph Koch, Sebastian Schaffert, and Sacha Berger. Querying the Web Reconsidered: Design Principles for Versatile Web Query Languages. *Journal of Semantic Web and Information Systems (IJSWIS)*, 1(2), 2005.
5. Sebastian Schaffert and François Bry. Querying the Web Reconsidered: A Practical Introduction to Xcerpt. In *Proceedings of the Conference Extreme Markup Languages*, 2004.
6. Sebastian Schaffert. *Xcerpt: A Rule-Based Query and Transformation Language for the Web*. Doctoral Thesis, Institute for Informatics, LMU, Munich, 2004.
7. François Bry and Sebastian Schaffert. Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. In *Proceedings of the International Conference on Logic Programming (ICLP)*, volume 2401 of *LNCS*. Springer-Verlag, 2002.
8. Pekka Kilpelainen and Heikki Mannila. Ordered and Unordered Tree Inclusion. *SIAM J. Comput.*, 24(2):348–356, 1995.
9. François Bry, Tim Furche, Clemens Ley, Bruno Marnette, Benedikt Linse, and Sebastian Schaffert. Datalog Relaunched: Simulation Unification and Value Invention. In *Proceedings of the Workshop Datalog 2.0, Oxford University*, 2011. To appear.
10. François Bry, Tim Furche, and Benedikt Linse. Simulation Subsumption or Déjà vu on the Web. In *Proceedings of the International Conference on Web Reasoning and Rule Systems*, LCNS, 2008.
11. Tim Furche. *Implementation of Web Query Language Reconsidered: Beyond Tree and Single-Language Algebras at (Almost) No Costs*. Doctoral Thesis, Institute for Informatics, University of Munich, 2008.
12. Sacha Berger. *Regular Rooted Graph Grammars - A Web Type and Schema Language*. Doctoral Thesis, Institute for Informatics, University of Munich, 2008.
13. François Bry and Klara Weiand. Flavours of KWQL, a Keyword Query Language for a Semantic Wiki. In *Proceedings of the 36th International Conference on Current Trends in Theory and Practice of Computer Science (SOSFEM)*, 2010.
14. Klara Weiand. *Keyword-Based Querying for the Social Semantic Web: The KWQL Language: Concept, Algorithm and System*. Doctoral Thesis, Institute for Informatics, University of Munich, 2011.
15. Klara Weiand, Fabian Kneißl, Tim Furche, and François Bry. PEST: Fast Approximate Keyword Search in Semantic Data using Eigenvector-based Term Propagation. Research Report, PMS-FB-2011-7 PMS-FB-2011-7, Institute for Informatics, University of Munich, 2011.
16. Jakub Kotowski, François Bry, and Simon Brodt. Reasoning as Axioms Change - Incremental View Maintenance Reconsidered. In *Proceedings of the 5th International Conference on Web Reasoning and Rule Systems (RR)*, volume 6902 of *LNCS*. Springer-Verlag, 2011.
17. Jakub Kotowski. *Constructive Reasoning for Semantic Wikis*. Doctoral Thesis, Institute for Informatics, University of Munich, 2011.
18. François Bry and Christoph Wieser. Web Queries with Style: Rendering Xcerpt Programs with CSS-NG. In *Proceedings of the 4th Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, 2006.
19. Michael Eckert, François Bry, Simon Brodt, Olga Poppe, and Steffen Hausmann. A CEP Babelfish: Languages for Complex Event Processing and Querying Surveyd. In Sven Helmer, Alex Poulovassilis, and Fatos Xhafa, editors, *Reasoning in Event-based Distributed Systems*. Springer-Verlag, 2011.
20. Paula-Lavinia Pătrânjan. *The Language XChange: A Declarative Approach to Reactivity on the Web*. Doctoral Thesis, Institute for Informatics, University of Munich, 2005.
21. Michael Eckert. *Complex Event Processing with XChange^{EQ}: Language Design, Formal Semantics and Incremental Evaluation for Querying Events*. Doctoral Thesis, Institute for Informatics, University of Munich, 2008.
22. Michael Eckert, François Bry, Simon Brodt, Olga Poppe, and Steffen Hausmann. Two Semantics for CEP, no Double Talk: Complex Event Relational Algebra (CERA) and its Application to XChange^{EQ}. In Sven Helmer, Alex Poulovassilis, and Fatos Xhafa, editors, *Reasoning in Event-based Distributed Systems*. Springer-Verlag, 2011.
23. Harald Zauner, Benedikt Linse, Tim Furche, and François Bry. A RPL through RDF: Expressive Navigation in RDF Graphs. In *Proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR)*, volume 6333 of *LNCS*, 2010.
24. Benedikt Linse. *Data Integration on the (Semantic) Web with Rules and Rich Unification*. Doctoral Thesis, Institute for Informatics, University of Munich, 2010.