Reasoning im und für das Semantic Web

Wolfgang May
Georg-August-Universität Göttingen
may@informatik.uni-goettingen.de

Zusammenfassung

Das "Semantic Web" bietet (und erfordert) ein weites Spektrum an wissenschaftlichen Teilbereichen, von der darunterliegenden technischen Realisierung über Datenhaltung und Wissensrepräsentation, Kommunikation bis hin zu –teilweise unentscheidbaren– Logikformalismen. Damit bietet es sich insbesondere zur bereichsübergreifenden, wenn man seine Anwendungen mit einbezieht, auch zur fachübergreifenden Forschung an. In diesem Beitrag wird ein Überblick gegeben wie in dem EU FP6 NoE "ReWeRSe"[1] (Reasoning on the Web with Rules and Semantics) Forschungsgebiete aus verschiedenen Bereiche der theoretischen und praktischen Informatik kombiniert werden. Hierbei werden insbesondere im Umfeld des Bereiches "Datenbanken" relevante Aspekte betrachtet.

1 Einführung und Überblick

Das *Semantic Web* bietet (und erfordert) ein weites Spektrum an Teilbereichen. Die Basis bilden einerseits Netzwerk- und *Kommunikationsaspekte*, von der darunterliegenden technischen Realisierung im ISO/OSI-Modell mit diversen Protokollen und speziellen Dienstleistungen bis hin zu Kommunikationsmechanismen auf abstrakter Ebene. Dazu kommen -in den einzelnen *Knoten* des Web, bzw. Semantic Web - Aspekte aus dem Bereich Datenbanken und Informationssysteme: Modellierung sowie Konzepte und Sprachen für Metadatendefinition, Anfragen an Daten und Metadaten, sowie Datenmanipulation. Der Schritt vom *Web-Knoten* zum *Web* kombiniert die genannten Bereiche, wobei Aspekte verteilter Datenbanken, Interoperabilität, Datenintegration, Sicherheit usw. hinzukommen. Mit der Hinzunahme *aktiver* bzw. *reaktiver* Funktionalität im Web, durch *Web Services* als auch Kommunikation zwischen Knoten (z.B. zur Propagierung von Updates) kommen Aspekte aus den Bereichen aktiver Datenbanken, insbesondere aktiver Regeln, weitere Sicherheitsaspekte, Vertrauen *(Trust)*, Verfahrensweisen (*Policies*), Spezifikation verteilter, dynamischer Prozesse etc. hinzu. Der weitere Schritt zum *Semantic Web* betrifft viele der obigen Aspekte (wobei manches dabei nicht unbedingt schwieriger, sondern durch die Nutzung von Semantik auch einfacher wird – z.B. Datenintegration):

Zum einen muss die Funktionalität der einzelnen beteiligten Knoten *lokal* semantik-tauglich erweitert werden. Hier spielen Ontologien und geeignete Modelle und Sprachen zur Wissensrepräsentation (sowohl extensional als auch intensional) mit zugrundeliegenden Formalismen/Logiken und Schlussmechanismen, sowie Anfragesprachen (Daten sowie auch Metadaten) eine Rolle. Zum zweiten muss diese Funktionalität innerhalb des *Semantic Web* integriert und "aktiviert" werden. Dies erfordert Modelle, die das Wissen mehrerer Knoten integrieren und damit die Grundlage für Anfrage "an das Semantic Web" bilden. Weiterhin müssen diese Modelle Änderungen sowohl lokal als auch global unter Berücksichtigung der Kommunikationsmechanismen ermöglichen. Um solche Änderungen operational zu realisieren sind entsprechende Sprachen zur Spezifikation und Umsetzung von Verhalten (einschließlich des Änderns von Verhaltensregeln) notwendig. Damit zeigt sich schlussendlich auch, dass "Anfragen" an das Semantic Web nicht nur einfache Anfragen sind, sondern ebenfalls von diesem Verhalten Gebrauch machen.

[1] http://rewerse.net

Um diese Anforderungen zu erfüllen muß nicht das Rad neu erfunden werden, sondern ein erfolgversprechender Weg ist, Erfahrungen und Konzepte unter anderem aus den Bereichen *Logik*, *Formale Methoden*, *Datenbanken*, *Kommunikation*, *Programmiersprachen* und *Softwareengineering* zu analysieren, weiterzuentwickeln, und zu kombinieren.

Da das Semantic Web keinerlei zentrale Struktur (weder topologisch, noch thematisch) besitzt, sondern einem "lebenden Organismus" bestehend aus sich autonom entwickelnden, kommunizierenden Knoten entspricht, ist insbesondere darauf zu achten, dass die entwickelten Konzepte modular sind, und *Konzepte* und die tatsächlichen *Sprachen* unabhängig, und dennoch kompatibel gestaltet werden um z.B. den Spezifika unterschiedlicher Anwendungsgebiete Rechnung zu tragen. Ein wichtiges Konzept, das in REWERSE schwerpunktmäßig verfolgt wird sind hierbei *deklarative*, insbesondere *regelbasierte* Sprachen.

Als Basis bietet XML und die damit zusammenhängenden Technologien ein inzwischen etabliertes Framework. Die zu entwickelnden Sprachen verwenden XML als Basis, d.h. arbeiten auf XML oder RDF (das häufig in XML-Repräsentation dargestellt wird) und besitzen auch selber eine XML-Syntax, die den Austausch von Daten, Wissen und Regeln ermöglicht. Die entwickelten Konzepte werden sowohl in eingeschränkten Demonstrator-Szenarios, als auch im praktischen Einsatz in ausgewählten Anwendungen in den Bereichen personalisierter Portale und Lernumgebungen [HN04] und Bioinformatik evaluiert.

Im folgenden wird zuerst kurz auf das zugrundeliegende Modell des Semantic Web eingegangen. In Abschnitt 3 werden die statischen Aspekte, d.h., Datenmodelle und Anfragesprachen, betrachtet bevor Abschnitt 4 die dynamischen Aspekte behandelt. Abschnitt 5 faßt die wichtigsten Punkte noch einmal zusammen und präsentiert einen Sprachvorschlag.

2 Architekturmodell

Wie oben geschrieben, wird das *Semantic Web* als ein System sich autonom entwickelnder, kommunizierender Knoten aufgefaßt:

- jeder Knoten verfügt über einen *lokalen* Zustand, gegeben durch extensionale Daten (Fakten), Metadaten (Schema, Ontologieinformationen), intensionale Daten (Regeln; *derivation rules*), sowie eine Verhaltensbasis (die in Abschnitt 4 betrachtet wird). Alle diese Komponenten können prinzipiell lokal autonom geändert werden.
- jeder Knoten kann Anfragen an andere Knoten stellen (die diese direkt beantworten, oder weitergeben) sowie Nachrichten von anderen Knoten empfangen (*peer-to-peer-Kommunikation*). Mit sochen Nachrichten können neben Antworten auf Anfragen auch Updates anderen Knoten mitgeteilt, bzw. auch ggf. der Zustand anderer Knoten geändert werden (geeignete Berechtigungen vorausgesetzt).
- es gibt Knoten die im wesentlichen als Datenquellen dienen (z.B. Fahrpläne oder Vorlesungsverzeichnisse), und Knoten, die im wesentlichen Dienstleistungen als "Infrastruktur" erbringen (Broker, pub/sub-Dienste, Continuous Query-Systeme etc.), sowie beliebige Zwischenformen (z.B. Reisebüro mit eigenen Daten sowie Broker-Funktionalität).

Die sich daraus für Anfragesprachen, Updates und *ECA-Regeln* ergebenden Konsequenzen sind in [MAB04] beschrieben.

3 Statisches Modell und Anfragen an Zustände

In dem betrachteten Modell des *Semantic Web* verfügt jeder Knoten über eine *lokale* Sicht, die seinen eigenen Zustand (einschließlich Metadaten, Ableitungs- und Verhaltensregeln sowie

der bisher erhaltenen Nachrichten, soweit gespeichert), sowie alle Daten, die ihm per Anfrage zugänglich sind, umfaßt.

Die lokale Sicht ist je nach eigenem Datenmodell z.B. durch eine Datalog-Datenbasis (mit diversen Interpretationsmöglichkeiten von Negation), durch eine F-Logic-Datenbank [KLW95] (das sich Klassenhierarchie, nichtmonotone Vererbung, einen flexiblen Schemabegriff, sowie Anfragen an Metadaten als Sprache für das Semantic Web anbietet), durch eine relationale oder XML-Datenbank, oder auch direkt als RDF/OWL-Daten [RDF00a, RDF00b, OWL04] gegeben, wobei die Datenmodelle von F-Logic und OWL bereits interne Reasoning-Mechanismen umfassen (bei OWL drei nach Ausdruckskraft und Komplexität/Entscheidbarkeit unterschiedliche Versionen "Lite", "DL", entsprechend *Description Logics*, und "Full").

[BM05] klassifiziert die verschiedenen Arten deduktiver Sprachen bzw. Schlussregeln, die hierfür benötigt werden, zum einen als Sichten (*konstruktive Regeln*), zum anderen zur Definition von Integritätsbedingungen und Wissen (*normative* und *deskriptive Regeln*). Konstruktive Regeln sind neben den bekannten Prolog/Datalog-Regeln auch im weiteren Sinne die SFW- bzw. FLWR-Konstrukte von SQL bzw. XQuery, sowie XSLT-Transformationen, die ja ebenfalls eine Sicht auf einen Datenbestand definieren. Normative Regeln können ebenfalls explizit als *Denials* gegeben sein, oder implizit z.B. in Form von DTDs, wobei hier der Üebergang zu *deskriptiven Regeln*, z.B. Ontologien, die auch wiederum unmittelbar als *konstruktive Regeln* zur Ableitung von Wissen genutzt werden können, fließend ist und die Einordnung im Detail auf die Intention und Verwendung durch den Benutzer ankommt.

Web-Fähigkeit der Knoten. Ein Knoten ist "passiv" Semantic-Web-fähig, wenn er nach aussen hin neben dem Zugriff auf Daten auch den Zugriff auf Metadaten und ihre Semantik in Form einer *Ontologie* ermöglicht. Dies geschieht derzeit üblicherweise per OWL, wobei die verwendete, anwendungsspezifische, Ontologie mit den Benutzern oder "Partnern" vereinbart sein muss.

Integrationsabbildungen. Dies ist in gewisser Weise ähnlich dem in verteilten bzw. föderierten Datenbanken [Con97] verfolgten Ansatz, ein gemeinsames Schema (hier: gemeinsame Ontologie) zu definieren. Eine Ontologie enthält jedoch neben dem reinen Schema weitere Metainformationen auf konzeptueller Ebene, z.B. über Beziehungen, abgeleitete Beziehungen etc. Aber, auch hier dient die gemeinsame Ontologie nur als *externes* Schema zwischen den beteiligten Knoten. Intern kann jeder Knoten unterschiedlich realisiert sein. In der Regel werden grosse Datenquellen selber keine RDF/RDFS/OWL-Daten enthalten sondern oft nicht einmal XML, sondern klassische relationale Datenbanken sein. Dieses lokale, logische Schema wird dann auf das globale (OWL-)Schema abgebildet. Zur Beschreibung von Abbildungen zwischen den lokalen und dem globalen (bzw. dem vereinbarten) Schema finden prinzipiell die in [Len02, Len03] diskutierten LAV/GAV-Konzepte ("global/local as view") bzw. GLAV (für P2P-Kommunikation) Anwendung.

"Globale" Anfragen und Ziehen von Schlüssen. Im Gegensatz zu der Situation bei föderierten Datenbanken, wo alle Teilnehmer explizit bekannt sind, können im Web grundsätzlich Teilnehmer "auftauchen" und "verschwinden". Es existiert –auch zu einer vereinbarten Ontologie– in der Regel keine zentrale Instanz, die alle Teilnehmer aufzählt (vgl. Hotels bei einem Reisebuchungs-Szenario). Neue Teilnehmer treten auf, indem sie "Informationen ins Web stellen" und versuchen, sich möglichst prominent referenzieren zu lassen. Im derzeitigen Web, und auch im *Semantic Web*, wird eine gewisse Organisation durch *Portale*, die auf Basis von für den Benutzer nicht sichtbarem Datenaustausch entlang explizit gespeicherter Beziehungen zwischen aktiven Knoten (entsprechend materialisierten Views bei "push"-Kommunikation, oder Unteranfragen bei "pull"-Kommunikation) die Informationen mehrerer Informationsquellen integrieren. Im derzeitigen Web geschieht dies oft noch durch individuelle Abbildungen der ein-

zelnen externen Schemata der Informationsquellen auf das interne Schema des Portals; für neue "Partner" muss somit jedes Mal eine Abbildung definiert werden. Ziel des *Semantic Web* ist, nur eine oder wenige Ontologien zu dem Themenbereich eines solchen Portals zu haben, die als externe Schemata der Informationsquellen zur Verfügung stehen. Tritt hierbei ein neuer Partner ein, muss er sich nur registrieren lassen. Ein Portal kann damit als Schnittstelle zu einer sich dynamisch umkonfigurierenden (sehr locker) föderierten Web-Fragment gesehen werden.

Falls innerhalb des betrachteten Web-Fragmentes unterschiedliche Ontologien verwendet werden, muss weiterhin eine Integrationsabbildung zwischen diesen verwendet (und zur praktischen Umsetzung auch in einem Vermittlungsservice implementiert) werden.

In dieser Situation können alle Anfragen also immer nur "nach bestem Wissen" von einem *Portal* beantwortet werden. Ein Portal besitzt damit ein über sein eigenes Wissen hinausgehendes *Modell*, das jedoch in mehreren Punkten von dem üblichen Modellbegriff abweicht:

- Konsistenz: die dem Portal verfügbaren Informationen können inkonsistent sein. Hier müssen entsprechende Mechanismen sowohl theoretisch als auch pragmatisch untersucht und angewendet werden.
- Unvollständigkeit: das Nicht-Bekanntsein einer Information bedeutet nicht, dass die Information in der Realwelt nicht zutrifft. Negative Schlüsse sind mit äußerster Vorsicht zu ziehen, und in der Regel als "es ist nicht bekannt, ob $p(x)$" zu interpretieren (was für den Benutzer dennoch häufig $\neg p(x)$ bedeutet, etwa wenn er ein Hotel buchen möchte).

Daraus ergibt sich, dass je nach Anwendung unterschiedliche, und insbesondere unterschiedlich aufwändige Logiken herangezogen werden (vgl. [BM05]): Übliche Anforderungen wie *excluded middle* ($A \vee \neg A$), *non-contradiction* ($\neg(A \wedge \neg A)$, Refutation $((A \rightarrow (B \wedge \neg B)) \rightarrow \neg A)$ sind nicht mehr gültig. Stattdessen benötigt man nichtmonotone Negation und disjunktive Theorien. Die o.g. Regeln müssen entsprechend dieser Semantiken interpretiert werden.

Auf die Kombination von Ontologien und (Schluss)regeln wird in [ADG+05] detailliert eingegangen (u.a., F-Logic, SWRL und DLP).

Anfragesprachen

Um sowohl der verteilten Natur der Informationen, als auch der zusätzlichen semantischen Ebene gerecht zu werden, müssen die Anfragesprache(n) sowohl beides unterstützen, als auch ggf. voneinander abgrenzen können:

- Anfragen an lokale Daten
- Anfragen an entfernte Daten gezielt per URL+Anfrage (auf der Ontologie-Ebene des *gemeinsamen* Schemas): Anfragen dieser Art werden insbesondere zur Propagation von Updates und sonstigen *Reaktionen* bei der in Abschnitt 4 behandelten Spezifikation von (lokalem und globalem) Verhalten verwendet. Da sie kein Web-weites Reasoning benötigen, sind sie effizienter auszuwerten.
- Anfragen an verteilte Daten auf Semantic-Web-Ebene. Auf dieser Ebene findet auch die Interaktion mit dem Benutzer statt.

Einen Überblick über existierende Anfragesprache für das Web und das Semantic Web wird in [FBS+04] gegeben; Anforderungen an Anfragesprachen sowie Updates sind in [MAB04] beschrieben.

Raum und Zeit. Spezielle Anforderungen an Modellierung, Reasoning und Anfragekonstrukte stellen die Bereiche *räumlicher* und *zeitlicher Daten*, wobei mit letzterem hier zeitliche Wertebereiche (wie etwa bei Terminkalendern), nicht temporale Anfragen über die Entwicklung der Datenbank gemeint sind. Neben typischen zeitbezogenen Anwendungen wie etwa Terminplanung und Auftrags-/Rechnungsabwicklung spielen zeitliche Annotationen auch eine Rolle, wenn die Aktualität von Nachrichten (Zeitpunkt des Auftretens eines Ereignisses gegenüber dem Zeitpunkt der Benachrichtigung) berücksichtigt werden muss.

4 Evolution und Reaktivität

Die beiden Aspekte *Evolution* und *Reaktivität* sind eng miteinander verbunden: Evolution kann –als Verhalten– z.B. durch reaktive Regeln deklarativ beschrieben und implementiert werden. Einen Überblick über beides findet man in [ABB+04, AM05], sowie eine Anforderungsanalyse für das Semantic Web in [MAB04].

Einfaches reaktives Verhalten ist bereits diesseits von Evolution bei der Beantwortung von Anfragen relevant:

- Reasoning kann durch deduktive oder reaktive Regeln beschrieben werden (vgl. die Äquivalenz des intern und in der originalen Spezifikation [KLW95] trigger-basierten Mechanismus für nichtmonotone Vererbung mit Default-Logik; [MK01]).
- Kommunikation zwischen verschiedenen Knoten zur Beantwortung von Anfragen. Reaktive Regeln können hier zur Implementierung der verschiedenen Strategien (*push*, *pull*, sowie *publish-subscribe* und *continuous queries*) verwendet werden [ABB+04, Kap. 1.7]).

Da zur verteilten Beantwortung von Anfragen weitere *Policies* angewendet werden (z.B. wenn eine Datenquelle nicht antwortet, oder um Konsistenz und *Trust* zu berücksichtigen), werden bereits in diesem Fall häufig reaktive Regeln Anwendung finden (für eine Bestandsaufnahme, siehe [BSD+05]).

Verteilte Anfragebearbeitung im Web. Um Anfragen auf Semantic-Web-Ebene effizient zu beantworten muss –wenn man nicht ein allgemeines Broadcasting machen will– zuerst geklärt werden, *welche* Partnerknoten relevante Antworten liefern können. Diese Auswahl basiert auf den Metadaten, die über die einzelnen Quellen bekannt sind [Kos00, Suc02, BDK+03].

Web Services beantworten oft nicht beliebige Anfragen in einer Anfragesprache, sondern nur eine eingeschränkte Menge von Formularanfragen. Ein solcher Web Service wird im Semantic Web üblicherweise durch eine Interfacebeschreibung in (z.B. in WSDL [WSD01] oder OWL-S [OWL03]) spezifiziert. Um solche Web Services zur (Teil)beantwortung von Anfragen zu nutzen, müssen erst geeignete Anfragen ausgesucht, und deren Antworten dann kombiniert werden (*Query Rewriting* [CGLV00, Hal01]).

4.1 Updates und Evolution im ("konventionellen") Web

Soweit wurden ein sich nicht veränderndes, und nur zur Anfragebearbeitung "aktives" Web betrachtet. Im gegenwärtigen Web tritt *Evolution* im wesentlichen in den folgenden einfachen Szenarien auf:

- Lokale Änderungen aufgrund externer Updates, oder lokale Evolution von Knoten als Reaktion auf einfache Ereignisse (etwa wie bei SQL-Triggern) auf Basis lokalen Wissens.
- Einfache Kommunikation (z.B. zur Propagation von Buchungen als Updates) zwischen Knoten durch *Nachrichten* und *reaktive Regeln*.

- Lokales Verhalten von *Web Services* als *Black Box*; die Ergebnisse werden ggf. durch Nachrichten mitgeteilt und auf die beiden vorhergehenden Fälle umgesetzt.
- Seltener: lokale Reaktionen auf lokale Ereignisse, die auch die *Sicht* des Knotens auf seine Umgebung (um nicht einfach von dem nicht existierenden "globalen" Modell zu sprechen) mit einbeziehen. Hierbei hat man erste Ansätze von *koordinierter* Evolution und Verhalten, womit z.B. weitergehende Konsistenz- und Plausibilitätsbedingungen berücksichtigt werden können.

4.2 Updates im Semantic Web

Die Sicht des *Semantic Web* als "lebender Organismus" bestehend aus sich autonom entwickelnden, kommunizierenden Knoten, der insgesamt ein "globales" Verhalten zeigt, führt zu einer Sichtweise als *kooperativer Evolution* der beteiligten Knoten. Neben lokalen Updates an einer Datenbasis müssen dazu verteilte Updates betrachtet werden:

- Updates an lokalen Daten müssen ggf. anderen Knoten, die diese Daten verwenden, zeitnah mitgeteilt werden. Hierzu muss festgestellt werden, wem welche Änderungen mitgeteilt werden müssen, womit prinzipiell dieselben Fragen wie bei *materialisierten Views* auftreten.
- *Intensionale Updates*, die sich auf abgeleitete, evtl. auch entfernte Daten beziehen. Dieser Fall entspricht einem *View Update*, das entsprechend weitergegeben werden muss.

In allen Fällen müssen die Integrations-Abbildungen vom lokalen auf das globale Schema hierzu "rückwärts" durchlaufen werden – d.h., egal ob diese auf LAV oder GAV basieren, endet man am Ende bei dem P2P-typischen GLAV.

Um das Szenario noch zu komplettieren, können auch unvollständig spezifizierte Updates betrachtet werden: Fakten können entweder durch Eingaben bekannt oder auf Ontologie-Level abgeleitet werden, die zu dem "bekannten" Zustand inkonsistent sind, etwa in Gegenwart unvollständiger Information, insbesondere aufgrund der zu erwartenden unvollständigen Nachrichtenübermittlung. Hierbei ist das Update (je nach Zuverlässigkeit) umzusetzen, und –evtl unter Einbeziehung weiterer Nachfragen an andere Knoten– wieder ein konsistenter Zustand herzustellen. Prinzipiell kann man dabei bis hin zu auf *Fuzzy Logics* oder *epistemischen Logiken* [KLM90] basierenden Ansätzen gehen.

Weiterhin kann man auch Updates der *deduktiven Regelbasis* (z.B. Evolving Logic Programs) oder –dem nächsten Abschnitt vorgreifend– Updates der Verhaltensbasis in Betracht ziehen (siehe [ABB+04]).

Update-Sprachen und -Konzepte. So wie Updates in Datenbanken auf den entsprechenden Anfragesprachen basieren, werden Update-Sprachen für das *(Semantic) Web* auf den entsprechenden Web-Anfragesprachen basieren. Verwendet man XML als internes Datenmodell, so ist dies z.B. XQuery+Updates; entsprechend RDQL+Updates für RDF-Daten, und entsprechende Konstrukte für Änderungen der Ontologien (womit man dem reinen Update auch sofort wieder entsprechendes Reasoning beiseitestellen muss, um die Konsistenz zu sichern).

4.3 Evolution und Verhaltensspezifikation

Die "globale", koordinierte Evolution im Semantic Web basiert auf geeignetem *Verhalten* der beteiligten Knoten. Hierbei bleibt die Reaktivität nicht nur auf Reaktionen auf einfache interne Ereignisse, Benutzerinteraktionen oder Nachrichten beschränkt, sondern umfasst auch komplexere Verhaltensweisen, etwa um anwendungsspezifisches Verhalten, z.B. *Business Rules* zu realisieren, oder auch *Policies* im Umgang mit eingehenden Informationen anzuwenden.

4.3.1 ECA-Regeln

Für eine zugleich deklarative und ausführbare Spezifikation von Verhalten bieten sich *reaktive Regeln* nach dem aus dem Bereich *aktiver Datenbanken* bekannten *Event-Condition-Action* (ECA)-Paradigma an [ABB+04, Kap. 4], [Pat99]. Kommunikation und reaktives Verhalten kann damit beides in einem auf der Basis von *Ereignissen (Events)* beschrieben werden: die beteiligten Knoten erkennen Ereignisse, überprüfen daraufhin eine *Bedingung (Condition)* und führen ggf. eine *Aktion (Action)* aus.

Eine ECA-Regel besteht entsprechend aus drei Teilen, die jeweils wiederum in einer Sub-Sprache für Ereignisse, Bedingungen (Anfragen), und Aktionen gegeben sind. Das Spektrum reicht dabei von einfachen reaktiven EA-Regeln bis hin zu komplexen Regeln deren Ausführung komplexe Ereigniserkennung, Anfragen, und Ausführung von Transaktionen beinhaltet.

ECA-Regeln können auf verschiedenen Abstraktionsebenen definiert (und implementiert) sein (siehe [ABB+05, Kap. 2]). Die tatsächliche Umsetzung der elementaren Ereigniserkennung geschieht auf der Datenbank-Ebene: Während für relationale Daten hier nur die bekannten SQL-Trigger zur Verfügung stehen, kann man bei XML-Daten entweder auf DOM-Ebene oder auf XPath/XQuery-Ebene ansetzen. Trigger auf der Ebene des Semantic Web spezifizieren ihre Ereignisse in der Terminologie des RDF bzw. OWL-Datenmodells. Intern müssen sie in den meisten Fällen auf XML- oder SQL-Trigger umgesetzt werden (wobei auch hier wieder berücksichtigt werden muss, dass die RDF/OWL-Schicht ein View ist).

Ereignisse. Ein (atomares) Ereignis ist allgemein jedes feststellbare Ereignis im Web, d.h., lokale Systemereignisse, ankommende Nachrichten (wobei sowohl der Nachrichteneingang selber ein Ereignis ist, als auch die Nachricht möglicherweise über das Eintreten eines anderen Ereignisses informiert), Transaktionsereignisse (Commit etc.), Updates von Daten, oder beliebige anwendungsspezifische Ereignisse. Neben diesen expliziten Ereignissen sollte es in Semantic-Web-Anwendungen auch möglich sein, Ereignisse abstrakt auf Ontologie-Level zu beschreiben; in diesem Fall muss ihre Erkennung auf geeignete Anfragen abgebildet (d.h. eine Zuordnung des Ereignisses zu einem oder mehreren elementaren Ereignissen in einem oder mehreren Knoten; ggf. muß auch eine Überwachung durch Continuous Queries stattfinden) und durchgeführt werden.

Reaktive Regeln beruhen nicht nur auf atomare Ereignissen, sondern verwenden häufig *zusammengesetzte Ereignisse (Composite Events)*, z.B. "wenn E_1 eintritt, und dann E_2 und E_3, aber nicht E_4 innerhalb höchstens 10 Minuten, dann führe A aus". Zusammengesetzte Events werden üblicherweise mit Hilfe von *Event Algebren* [ABB+04, Kap. 2.7], [CKAK94] beschrieben. Die Erkennung von (zusammengesetzten) Ereignissen kann Parameter einzelner Ereignisse an Variablen binden und zurückgeben.

Bedingungen. Bedingungen sind üblicherweise Anfragen. Sie können Parameter enthalten, die durch die Ereignisse definiert wurden, und auch neue Variablen binden.

Aktionen. Aktionen sind häufig einfache Updates oder das Senden on Nachrichten, können aber auch durch kompliziertere Spezifikationen gegeben sein. Sie können mit (aus der Ereigniserkennung und der Auswertung der Bedingung gebundenen) Variablen parameterisiert sein. Der Aktions-Teil kann z.B. als Ausdruck einer Prozessalgebra (CCS/CSP) oder Term über den Aktionen einer Aktionslogik gegeben sein.

Globale ECA-Regeln. Die oben beschriebenen Konzepte gehen implizit von einer *lokalen* Auswertung der ECA-Regeln in einem Knoten aus. Weitergehend können auch *globale* Regeln definiert werden, die nicht mehr ausschliesslich lokal ausgewertet werden können:

- Ereignisse, die explizit verschiedenen Knoten zugeordnet sind,

- *intensionale*, abstrakt definierte Ereignisse für die die genaue Zuordnung zu einem Knoten nicht angegeben ist, und die auch oft nicht auf ein einzelnes Update abbildbar sind, sondern ggf. erst abgeleitet werden müssen. Zur tatsächlichen Erkennung des Ereignisses bleibt hier wohl oft nur der Umweg über *Continuous Queries* übrig.

Globale ECA-Regeln können z.B. als Dienst von Portalen ausgewertet werden.

Auswertung von ECA-Regeln. Der theoretisch interessanteste Teil ist hier die Erkennung von zusammengesetzten Ereignissen. Ein naiver Ansatz wäre, alle Ereignisse zu speichern, und zusammengesetzte Ereignisse als Anfragen an diese *Event Base* umzuschreiben. Dies ist aus Effizienzgründen nicht praktikabel (nach jedem atomaren Ereignis müssten alle Anfragen komplett ausgewertet werden). Stattdessen wird die Ereigniserkennung *inkrementell* durchgeführt: jeder zu erkennende zusammengesetzte Ereignisausdruck wird auf einen Automaten-Schema abgebildet. Wird ein atomares Ereignis festgestellt, wird entsprechend ein Automaten-Schema instanziiert (mit den aktuellen Parametern) und alle bereits "laufenden" Automaten ggf. weitergeschaltet (äquivalente Formalismen existieren auf Basis von Graphen sowie temporallogischen Formeln).

Weiterhin ist es wünschenswert, innerhalb zusammengesetzter Ereignisse auch bereits Werte und Bedingungen aus dem gegenwärtigen Zustand auszuwerten. Bei Verwendung einer inkrementellen Ereigniserkennung sind diese Tests kein Problem.

Neben ECA-Regeln existieren weitere in Frage kommende Formalismen, z.B. Prozessalgebren (CCS/CSP) [Mil83, Hoa85] oder die –ebenfalls regelbasierte– *Transaction Logic* [BK94], die gleichzeitig zur Planung und zur Ausführung von Aktionen verwendet werden kann.

Da die Semantik von Transaction Logic –im Gegensatz zu ECA-Regeln– direkt auf den Begriffen *Struktur* und *Formeln* und einer *Modelltheorie* mit Tarski-Semantik [Kei78] basiert, eignet sich dieses Framework auch direkt um (Korrektheits)Aussagen *über* Regeln und Zustände zu machen. Andere logikbasierte Ansätze verwenden "klassische" modale Temporallogik und Kripke-Strukturen [Eme90] zur Verifikation.

4.3.2 ECA-Sprachentwurf

Um die beschriebenen Anforderungen zu erfüllen, müssen mehrere (immer noch generische) Sprachen und Teilsprachen definiert werden: Zumindest wird eine umgebende Sprache für ECA-Regeln benötigt, die eine Sprache für zusammengesetzte Ereignisse (die wiederum mit atomaren Ereignissen parametrisiert ist), eine Sprache für Bedingungen und eine Sprache für Aktionen einbettet (und eine Variablenübergabe mit diesen ermöglicht). In Anbetracht der Tatsache, dass es jeweils viele verschiedene Vorschläge (und wie oben gesehen auch Abstraktionsebenen) für Event-Algebren, Anfragesprachen und auch für Aktionsspezifikationssprachen gibt, sollte man jeweils nicht nur die Einbettung einer einzelnen, gegebenen Sprache vorsehen, sondern einen modularen Ansatz wählen, der es ermöglicht –unter gewissen Bedingungen– beliebige Sprachen einzubetten.

5 Entwicklung Regelbasierter Sprachen für das Semantic Web

In [BM05] werden Anforderungen an die zu entwickelnden logikbasierten Mechanismen und Sprachen für das Semantic Web definiert, von denen einige hier bereits genannt wurden und deren Quintessenz zum Schluss noch einmal zusammengefasst wird:

- formales, logik-basiertes (Daten)Modell und Modelltheorie mit modular aufgebauter Reasoning-Funktionalität (Negation, Inkonsistenz, Unvollständigkeit),

- auf dem Datenmodell aufbauende Anfragesprache(n) (sowohl Anfragen an den Zustand, als auch im weiteren Sinne die Sprache zur Modellierung von Ereignissen) mit deklarativer Semantik,
- Sprache zur Spezifikationen von Aktionen muss zumindest eine operationale Semantik besitzen; wünschenswert wäre auch hier eine modelltheoretische Semantik,
- alle Sprachen müssen getypt sein,
- Modularität in Syntax und Darstellung, Semantik und Auswertung/Ausführung der Sprachen und Sprachkonzepte: Interoperabilität und Komponierbarkeit (*Composability*).

Kohärenz in Syntax und Darstellung wird u.a. dadurch erreicht, dass alle Sprachen eine XML-Repräsentation (vgl. XSLT, XML Schema) besitzen. Interoperabilität und Komponierbarkeit erfordert wohldefinierte Schnittstellen, zu denen wiederum die durchgehende Typisierung beitraägt. Weiterhin wird dies durch die Definition einer gemeinsamen, erweiterbaren Ontologie unterstützt, womit die Sprachen auch selber wieder zum Teil des Semantic Web werden (und die einzelnen Regeln zu Objekten im Semantic Web).

Sprachvorschlag (Entwurf). Das untenstehende Beispiel zeigt einen groben Sprachvorschlag für eine Markup-Language "ECA-ML" [ABB+05] für ECA-Regeln, der deutlich an XSLT angelehnt ist. Jede Regel (und auch andere Konstrukte) kann Variablen definieren und mit Werten initialisieren. Eine Regel besteht aus je einem eca:event, eca:condition (optional) und eca:action-Element. Jedes dieser Elemente besitzt ein eca:name-Attribut, sowie ein href-Attribut, das auf eine URI verweist. Ist diese URI ein Directory, so könnte dort z.B. eine Beschreibung der jeweiligen Sprache als XML-Schema oder OWL-Instanz liegen, oder auch im Fall einer Event-Sprache ein Java-Klasse oder ein Web Service, die den Detektionsalgorithmus implementiert (oder eine einfachere Menge von ECA-Regeln, die den Automaten beschreiben) und vom ECA-Interpreter bei Bedarf geladen oder aufgerufen werden kann.

Der Event-Teil der Regel ist als Folge spezifiziert, deren erstes atomares Ereignis das Löschen eines Elements der Antwortmenge von $xpath\text{-}expr_1$ der lokalen Datenbasis ist (ähnlich dem XSLT-Mechanismus). Beim Eintreten eines solchen Events wird die Variable var_1 an das gelöschte Element e gebunden, und die Variable var_2 an $e/xpath\text{-}expr_3$. Wenn danach das im Beispiel nicht näher spezifizierte Teilevent schrittweise detektiert wird, ist am Ende der Ereignis-Teil der Regel erfüllt. Als nächstes wird der eca:condition-Teil ausgewertet (der hier nur einen XPath-Ausdruck testet) – hier könnten ggf. vorher gebundene Variablen verwendet und weitere Variablen gebunden werden. Ist die Beduingung erfüllt, wird der eca:action-Teil ausgeführt (ggf. werden hier die vorher gebundenen Variablen verwendet).

```
<eca:rule>
  <eca:variable name="...">xpath</variable>
  <eca:event name= "eca-events-basic" href="uri" >
    <evt:seq>
      <evt:atomic>
        <delete-of select="xpath-expr1" >
          <variable name="var1" select="." />
          <variable name="var2" select="xpath-expr3" />
        </change-of>
      </evt:atomic>
      <evt:...>
        Spezifikation eines weiteren (zusammengesetzten) Events
      </evt:...>
```

```
    </evt:seq>
  </eca:event>
  <eca:condition language="XPath" href="uri">
    xpath-expr
  </eca:condition>
  <eca:action language="XQuery+Updates" href="uri">
    update xpath-expr_4
    set xpath-expr_5 := value
  </eca:action>
</eca:rule>
```

Acknowledgement. This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`).

Literatur

Die in der folgenden Literaturliste aufgeführten Referenzen bieten einen umfassenden Überblick über den State of the Art in den behandelten Bereichen.

[ABB+04] José Júlio Alferes, James Bailey, Mikael Berndtsson, Franois Bry, Jens Dietrich, Alexander Kozlenkov, Wolfgang May, Paula-Lavinia Pătrânjan, Alexandre Pinto, Michael Schröder, and Gerd Wagner. State-of-the-art on evolution and reactivity. Technical Report I5-D1, REWERSE EU FP6 NoE, 2004. Available at `http://www.rewerse.net`.

[ABB+05] José Júlio Alferes, Mikael Berndtsson, François Bry, Michael Eckert, Wolfgang May, Paula Lavinia Pătrânjan, and Michael Schröder. Use cases in evolution and reactivity. Technical Report I5-D2, REWERSE EU FP6 NoE, 2005. Available at `http://www.rewerse.net`.

[ADG+05] Grigoris Antoniou, Carlos Viegas Damsio, Benjamin Grosof, Ian Horrocks, Michael Kifer, Jan Maluszynski, and Peter F. Patel-Schneider. Combining Rules and Ontologies. A survey. Technical Report I3-D3, REWERSE EU FP6 NoE, 2005. Available at `http://www.rewerse.net`.

[AM05] José Júlio Alferes and Wolfgang May. Course: Evolution and reactivity for the web. In *REWERSE Summer School*, 2005. to appear with Springer LNCS.

[BDK+03] Ingo Brunkhorst, Hadhami Dhraief, Alfons Kemper, Wolfgang Nejdl, and Christian Wiesner. Distributed queries and query optimization in schema-based p2p-systems. In *Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pages 184–199, 2003.

[BK94] A. J. Bonner and M. Kifer. An overview of transaction logic. *Theoretical Computer Science*, 133(2):205–265, 1994.

[BM05] François Bry and Massimo Marchiori. Ten Theses on Logic Languages for the Semantic Web. In *Proceedings of W3C Workshop on Rule Languages for Interoperability*, 2005.

[BSD+05] Piero A. Bonatti, Nahid Shahmehri, Claudiu Duma, Daniel Olmedilla, Wolfgang Nejdl, Matteo Baldoni, Cristina Baroglio, Alberto Martelli, Viviana Patti, Paolo Coraggio, Grigoris Antoniou, Joachim Peer, and Norbert E. Fuchs. Rule-based policy

specification - state of the art and future work. Technical Report I3-D3, REWERSE EU FP6 NoE, 2005. Available at `http://www.rewerse.net`.

[CGLV00] Diego Calvanese, Guiseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is query rewriting? In *Knowledge Representation meets Databases (KRDB 2000)*, pages 17–27. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-29/`, 2000.

[CKAK94] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proceedings of the 20th VLDB*, pages 606–617, 1994.

[Con97] Stefan Conrad. *Föderierte Datenbanksysteme: Konzepte der Datenintegration*. Springer, 1997.

[Eme90] E. A. Emerson. Temporal and modal logic. volume B: Formal Models and Semantics, chapter 16, pages 995–1073. Elsevier, 1990.

[FBS+04] Tim Furche, François Bry, Sebastian Schaffert, Renzo Orsini, Ian Horrocks, Michael Krauss, and Oliver Bolzer. Survey over Existing Query and Transformation Languages. Technical Report I4-D1, REWERSE EU FP6 NoE, 2004. Available at `http://www.rewerse.net`.

[Hal01] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[HN04] Nicola Henze and Wolfgang Nejdl. A Logical Characterization of Adaptive Educational Hypermedia. *New Review of Hypertext and Hypermedia (NRHM)*, 10(1):77–113, 2004. Available at `http://www.kbs.uni-hannover.de/Stamm/publikationen/publikationen.html`.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[Kei78] J. Keisler. Fundamentals of model theory. In *Handbook of Mathematical Logic*, pages 47–103. North Holland, 1978.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[Kos00] Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[Len02] Maurizio Lenzerini. Data integration: a theoretical perspective. pages 233–246, 2002.

[Len03] Maurizio Lenzerini. Tutorial on information integration. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[MAB04] Wolfgang May, José Júlio Alferes, and François Bry. Towards generic query, update, and event languages for the Semantic Web. In *Principles and Practice of Semantic Web Reasoning (PPSWR)*, number 3208, pages 19–33. Springer, 2004.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, pages 267–310, 1983.

[MK01] Wolfgang May and Paul-Th. Kandzia. Nonmonotonic inheritance in object-oriented deductive database languages. *Journal of Logic and Computation*, 11(4), July 2001.

[OWL03] OWL-S: Web Service Ontology. `http://www.daml.org/services/owl-s/`, 2003.

[OWL04] OWL Web Ontology Language. `http://www.w3.org/TR/owl-features/`, 2004.

[Pat99] N. W. Paton, editor. *Active Rules in Database Systems*. Monographs in Computer Science. Springer, 1999. ISBN 0-387-98529-8.

[RDF00a] Resource Description Framework (RDF). `http://www.w3.org/RDF`, 2000.

[RDF00b] Resource Description Framework (RDF) Schema specification. `http://www.w3.org/TR/rdf-schema/`, 2000.

[Suc02] Dan Suciu. Distributed query evaluation on semistructured data. *ACM Transactions on Database Systems*, 27(1):1–62, 2002.

[WSD01] Web Services Description Language (WSDL). `http://www.w3.org/TR/wsdl`, 2001.