Xcerpt und visXcerpt: deduktive Anfragesprachen für das Web

Sacha Berger, François Bry, Sebastian Schaffert
Institut für Informatik, Ludwig-Maximilians-Universität München
http://www.pms.ifi.lmu.de

1 Übersicht

Mit der zunehmenden Bedeutung von XML als Format für den Datenaustausch und zur Repräsentation von semistrukturierten Datenbanken steigt auch das Interesse an Anfrage- und Transformationssprachen für XML und semistrukturierte Daten.

Viele solcher Sprachen, wie XPath und XQuery sind *navigationell* insofern, als dass ihr Ansatz zur Variablenbindung vom Programmierer verlangt, explizite Pfadnavigationen durch Dokumente oder Datensätze anzugeben. Im Gegensatz dazu sind einige andere Anfragesprachen „musterbasiert": ihr Ansatz zur Variablenbindung entspricht dem der mathematischen Logik, das heißt der Programmierer gibt Muster oder Schablonen an, die an ein oder mehreren Stellen Variablen enthalten können. Ein musterbasierter Ansatz zur Variablenbindung kann daher das Schreiben und Lesen komplexer Anfragen erleichtern und somit die Effizienz der Programmierung erhöhen. Eines der Ziele dieses Artikels ist, diese Behauptung anhand praktischer Beispiele zu belegen.

Xcerpt [1] ist eine experimentelle, regel- und musterbasierte Anfragesprache für XML und semistrukturierte Daten. Xcerpt benutzt Muster sowohl um in Anfrageausdrücken Variablen zu binden, als auch dazu, die Variablenbindungen (die durch die Auswertung einer Anfrage entstehen) mit Hilfe von so genannten Konstruktionstermen in neuen Strukturen wieder zusammenzufügen. Ein weiteres Ziel dieses Artikels ist zu zeigen, dass eine musterbasierte Ergebniskonstruktion kombiniert mit einem musterbasierten Ansatz zur Variablenbindung zu einer intuitiven, benutzerfreundlichen und effizient zu programmierenden Anfragesprache führt.

Xcerpt ist insofern experimentell, als dass es als Ziel hat, einen anderen, nicht-navigationellen Ansatz zur Anfrage von Web-Daten zu untersuchen, als denjenigen der am weitesten verbreiteten Anfragesprachen XPath und XQuery. Nichtsdestotrotz existiert eine prototypische Implementierung der Sprache, die derzeit verwendet wird um Anwendungsszenarien zu entwickeln.

Eine weitere wichtige Eigenschaft von Xcerpt ist der Regelansatz: Xcerpt-Anfragen bestehen aus Regeln, die den `VIEWS` in SQL sehr ähnlich sind, aber im Gegensatz zu SQL sind auch Verknüpfung von Regeln und Rekursion möglich. Regeln können zur besseren Strukturierung von komplexen Anfragen dienen und daher durch die Aufteilung in kleinere, leichter verständliche Einzelschritte – ähnlich wie Prozeduren in imperativen Programmiersprachen – die Programmierung erleichtern. Diese dritte Behauptung zu belegen ist ein weiteres Ziel dieses Artikels.

Aufgrund des regelbasierten Ansatzes wurde Xcerpt im Rahmen des EU Network of Excellence REWERSE (`http://rewerse.net`) als Basis für die weitere Entwicklung von Inferenzsprachen für das Web gewählt.

Aufbauend auf Xcerpt wurde eine *visuelle Anfragesprache* mit Namen *visXcerpt* entworfen und implementiert. Interessanterweise scheint sich der muster- und regelbasierte Ansatz von Xcerpt besonders gut als Basis für eine visuelle Darstellung von Anfragen und Transformationen zu eignen. Der Grund hierfür ist möglicherweise, dass Muster bereits zweidimensionale Strukturen sind, die sehr ähnlich zur visuellen Darstellung sind. Den meisten visuelle Anfragesprachen für XML und/oder semistrukturierte Daten (wie z.B. XML-GL [2], GraphLog [3], VXT [4], BBQ [5] und Xing [6]) sowie die Sprache QBE und diverse Verbesserungen (z.B. MS Access) liegt ein musterbasierter Ansatz zu Grunde. Diese vierte Behauptung zu belegen ist das letzte Anliegen dieses Artikels.

Interessanterweise, und möglicherweise stützt das die zuletzt genannte Behauptung, kann eine visuelle Anfragesprache, die auf einer textuellen, muster- und regelbasierten Anfragesprache aufbaut, Regeln einfach visuell darstellen (z.B. mit Hilfe eines Stylesheets), anstatt komplexe Transformationen durchzuführen.

2 Positionelle und Navigationelle Datenselektion

Von großer Bedeutung für die Anfrage von semistrukturierten Daten ist die Art der Selektion von Datensätzen innerhalb eines Dokuments. Den meisten weitverbreiteten Anfragesprachen – z.B. XQuery – liegt eine pfadbasierte Selektion, ausgedrückt in XPath (oder ähnlichen Ansätzen), zu Grunde. XPath stellt Konstrukte wie reguläre Pfadausdrücke und Wildcards zur Verfügung, die es erlauben, Pfade in einem Baum oder Graphen anzugeben. Zum Beispiel hat der XPath-Ausdruck `/a[b]//c` die Bedeutung „finde die Knoten mit Namen c die von der Wurzel aus erreicht werden können über a-Knoten, welche einen direkten Nachfolger mit Namen b haben und auf beliebiger Tiefe die mit c bezeichneten Knoten enthalten". Eine derartige Datenselektion kann als *navigationell* bezeichnet werden.

Für einfache Anfragen und Transformationen ist der navigationelle Ansatz sehr natürlich und Programme sind einfach zu schreiben. Wenn jedoch kompliziertere Anfragen benötigt werden, besonders, wenn mehrere Datensätze ausgewählt werden müssen, führt der navigationelle Ansatz jedoch oft zu schwer verständlichen, und daher auch schwer zu schreibenden, Programmen. Die Vermischung von Anfrage und Konstruktion in Anfragesprachen wie XQuery trägt zusätzlich zu diesen Schwierigkeiten bei.

Außerdem erscheinen wegen der Möglichkeit in XPath sowohl Vorwärts- als auch Rückwärtsachsen anzugeben, Anfragen oft unnötig kompliziert, obwohl ihre eigentliche Bedeutung sehr einfach ist.

Ein weiterer wesentlicher Aspekt der navigationellen Selektion ist, dass es nicht oder nur schwer möglich ist, mehrere zusammengehörige Knoten gleichzeitig auszuwählen. Eine solche Auswahl wäre jedoch sehr natürlich und wird in beinahe allen komplizierteren Anfragen benötigt, wie zum Beispiel in einer Literaturliste, in der gleichzeitig die Autoren, der Titel und das Erscheinungsjahr einer Veröffentlichung ausgewählt werden sollen. Jeder, dem Literaturlisten geläufig sind, hat sofort das Bild der Datenbank und einer passenden Anfrage vor Augen, die der Struktur der Datenbank entspricht und in der die Variablen an entsprechenden Positionen stehen. Musterbasierte, oder *positionelle*, Anfrage- und Transformationssprachen wie Xcerpt unterstützen eine solche intuitive "Visualisierung".

In der positionellen Anfragesprache Xcerpt [1] werden die auszuwählenden Knoten durch Variablen in so genannten *Anfragetermen* angegeben. Analog dazu wird die Konstruktion des Ergebnisses durch so genannte *Konstruktionsterme* festgelegt, die durch das Vorkommen der gleichen Variablen mit Anfragetermen verknüpft sind. Konstruktionsterme werden durch *Regeln* mit Konjunktionen/Disjunktionen von Anfragetermen in Zusammenhang gebracht.

Der positionelle Ansatz zur Anfrage von semistrukturierten Daten wurde als Erstes in den Anfragesprachen UnQL [7] und XML-QL [8] untersucht. Beide Sprachen unterscheiden sich jedoch von Xcerpt dadurch, dass sie nicht das Verknüpfen mehrerer Regeln erlauben. Allgemein findet der positionelle Ansatz seine Wurzeln in der funktionalen und logischen Programmierung. Außerdem können die beiden Anfragesprachen QBE und SQL als positionell angesehen werden.

3 Wesentliche Elemente von Xcerpt

Ein Xcerpt-Programm besteht aus mindestens einem *Ziel* (engl. *goal*) und 0 oder mehreren *Regeln* (engl. *rules*). Ziele und Regeln sind zusammengesetzt aus Daten-, Anfrage- und Konstruktionstermen (engl. data, construct and query terms), die zunächst eingeführt werden sollen. Neben der "abstrakten" Termsyntax, die hier vorgestellt wird, gibt es für Xcerpt auch eine XML-Syntax, die aus Platzgründen hier nicht beschrieben ist.

3.1 Daten-, Anfrage- und Konstruktionsterme

Allen Termen gemeinsam ist, dass sie baum- oder graphartige Strukturen darstellen. Die Nachfolger (oder "Kinder") eines Knotens können entweder *geordnet* (wie in XML) oder *ungeordnet* (wie in den meisten Datenbanken) sein. Eckige Klammern geben an, dass die Kinder eines Terms geordnet sind, d.h. die entsprechenden Teilterme der Datenbank haben die gleiche Reihenfolge wie die Teilterme des Anfrageterms (*geordnete* Termspezifikation). Geschweifte Klammern geben an, dass die Kinder eines Terms ungeordnet sind, d.h. die entsprechenden Teilterme können in beliebiger Reihenfolge in der Datenbank vorkommen (*ungeordnete* Termspezifikation).

Einfache Klammern werden verwendet, um auszudrücken, dass ein passender Datenterm außer den den Teiltermen der Anfrage entsprechenden Teiltermen keine weiteren Teilterme enthalten darf. (*totale* Termspezifikation). Doppelte Klammern werden verwendet, um auszudrücken, dass der Datenterm zusätzliche Teilterme enthalten darf, solange für jeden Teilterm der Anfrage ein entsprechender Teilterm im Datenterm vorhanden ist (*partielle* Termspezifikation).

Datenterme (engl. *data terms*) werden zur Darstellung von XML-Dokumenten und den Datensätzen semistrukturierter Datenbanken verwendet. Sie ähneln den variablenfreien Ausdrücken in funktionalen Programmiersprachen und logischen Atomen. Eine *Datenbank* ist eine (Multi-)Menge von Datentermen (z.B. das Web).
Beispiel: Bücherdatenbank

```
bib { book {
    title { "Data on the Web" },
    authors [ author { "Serge Abiteboul" },
              author { "Peter Buneman" },
              author { "Dan Suciu" } ],
    price { "69.95" } }, ...
}
```

Anfrageterme (engl. *query terms*) ähneln Ausdrücken in funktionalen Programmiersprachen oder logischen Atomen, die Variablen enthalten können. Im Gegensatz zu diesen haben Anfrageterme jedoch folgende Eigenschaften:

- in einem Anfrageterm ist es möglich, partielle Termspezifikationen zu verwenden, um irrelevante Teile auszulassen,
- in einem Anfrageterm können, wie Datenbanktermen, Teilterme geordnet oder ungeordnet sein,
- in einem Anfrageterm ist es möglich, Teilterme auf beliebiger Tiefe anzugeben (*descendant*).

Beispiel: Folgender Anfrageterm wählt Titel und Autoren aus:

```
bib {{ book {{
    var T ⇝ title, authors {{ var A }}
}} }}
```

Um Bindungen für die in einem Anfrageterm vorkommenden Variablen zu erhalten, werden Anfrageterme mit Daten- oder Konstruktionstermen *unifiziert*. Xcerpt verwendet einen eigenen Unifikationsalgorithmus namens *Simulationsunifikation* [9], der im Gegensatz zu den herkömmlichen Unifikationsalgorithmen die Eigenschaften von XML (wie heterogene Daten, ungeordnete Kindelemente, usw) besser berücksichtigt. Simulationsunifikation basiert auf *Graphsimulation* [10], einer Relation ähnlich zu Graphhomomorphismen.

Das Ergebnis der Unifikation eines Anfrageterms mit einem Datenterm ist eine Menge von Substitutionen für die im Anfrageterm vorkommenden Variablen. Die Anwendung jeder dieser Substitutionen aif den Anfrageterm ergibt einen variablenfreien Anfrageterm, der im Sinne der Graphsimulation in den Datenterm simuliert.

Das Xcerpt-Konstrukt $X \rightsquigarrow t$ (sprich: "as") dient dazu, ein Variablenvorkommen durch einen Anfrageterm zu beschränken, so daß nur solche Bindungen möglich sind, mit denen der Anfrageterm unifiziert. Das Xcerpt-Konstrukt *desc* (sprich: "descendant") kann benutzt werden, um Teilterme auf beliebiger Tiefe anzugeben.

Konstruktionsterme dienen dazu, aus Anfragetermen stammende Variablenbindungen in einer neuen Struktur anzuordnen, um neue Datenterme zu erzeugen. In einem Konstruktionsterm können die Konstrukte { } und [ ], sowie Variablen vorkommen, aber $\rightsquigarrow$-Einschränkungen sind nicht zulässig. Der Grund für diese Beschränkung ist, daß Variablenbindungen in Anfragetermen definiert werden, während in Konstruktionstermen die Bindungen nur gesammelt werden. Diese strikte Trennung zwischen Anfrage und Konstruktion erscheint vorteilhaft.

Ausserdem können Konstruktionsterme das Konstrukt *all* t enthalten, welches dazu dient, alle Instanzen von t die sich aus verschiedenen Substitutionen für die Variablen in t ergeben können, aufzusammeln.

Beispiel: Der folgende Konstruktionsterm sammelt alle Titel/Autor Paare aus der vorangehenden Anfrage:

```
results { all result { var T, var A } }
```

3.2 Xcerpt-Regeln

Eine Xcerpt-Regel setzt einen Konstruktionsterm mit einer Anfrage in Beziehung, die aus UND und/oder ODER-Verknüpften Anfragetermen bestehen kann. Eine Xcerpt-Regel hat die Form

```
CONSTRUCT Konstruktionsterm FROM Anfrage END
```

Eine solche Regel kann als eine “Sicht” (engl. “View”) auf XML Daten von möglicherweise verschiedenen Quellen gesehen werden, die neue Dokumente in der Form des Konstruktionsterms erzeugt indem die Anfrage gegen die Web-Ressourcen ausgewertet wird.

Eine Xcerpt-Anfrage kann eine oder mehr Referenzen auf externe *Ressourcen* enthalten, die angefragt werden sollen. Desweiteren können Xcerpt-Regeln wie in deduktiven Datenbanken oder Logikprogrammen über *chaining* verknüpft werden, um komplexe Anfragen zu modellieren.

3.3 Weitere Konstrukte

Die vorangegenden Abschnitte beschreiben nur die wichtigsten Konstrukte der Sprache Xcerpt. Zusätzlich stellt Xcerpt noch folgende fortgeschrittene Konstrukte zur Verfügung:

- Referenzmechanismus um Graphstrukturen zu repräsentieren
- grundlegende Operationen und Aggregationen auf Basistypen wie *int* oder *string*
- Negation
- reguläre Ausdrücke für die Anfrage von Text
- Positionsangaben für die Selektion anhand der Position

4 visXcerpt: Eine Visuelle Darstellung von Xcerpt

Entscheidend für die visuelle Sprache visXcerpt ist, dass Xcerpt eine termbasierte Sprache ist, die für jedes Variablenvorkommen die *Position* anstelle einer *Navigation* (beginnend von der Wurzel zum Vorkommen der Variablen) in einem Anfrage- oder Konstruktionsausdruck angibt. Infolgedessen kann man Xcerpts visuelles Gegenstück *visXcerpt* einfach als andere *Darstellung* auffassen statt als völlig neue Sprache. Diese Darstellung ist nichts anderes als ein fortgeschrittenes Layout mit einigen dynamischen Aspekten. VisXcerpt ist implementiert mit Hilfe von CSS Stylesheets für das statische Layout und ECMA Script für dynamische Eigenschaften.

Xcerpt-Terme werden als Boxen dargestellt. Ein Termbezeichner (“Tagname”) wird als Register an eine Box angehängt. VisXcerpt unterstützt sowohl Attribute als auch Textinhalt. Attribute werden in einer zweispaltigen Tabelle festgelegt, die die Attributsnamen in der linken Spalte und die Attributswerte in der rechten Spalte enthält. Die Attributstabelle ist immer das erste Element einer Box und wird weggelassen, falls es keine Attribute gibt. Unmittelbare Teilterme (d.h. Kinder) werden gleichermaßen als verschachtelte Boxen dargestellt. Verschachtelte Boxen werden vertikal angeordnet. Zur besseren Unterscheidung erhalten sie verschiedene Farben.

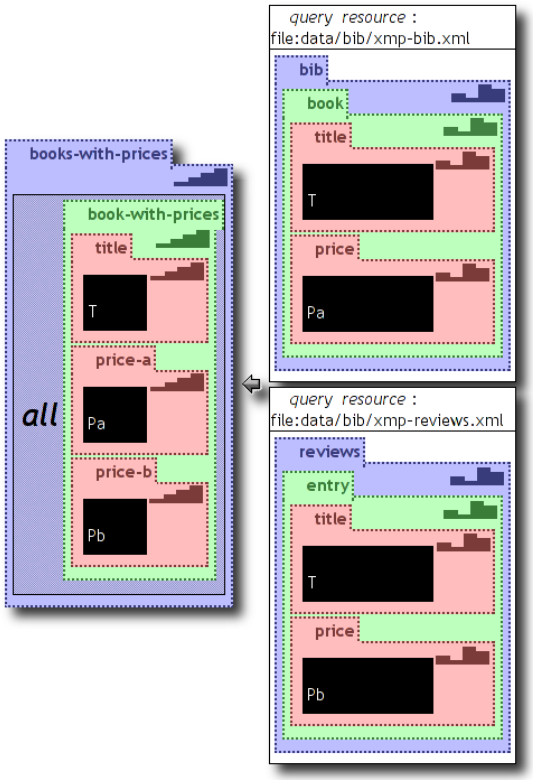

Abbildung 1: Eine Regel in der visuellen Syntax von Xcerpt

Um zwischen totalen und partiellen Termangaben zu unterscheiden, verwendet visXcerpt durchgezogene bzw. gestrichelte Ränder. Die Termeigenschaften “geordnet” und “ungeordnet” werden durch ein Icon in der rechten oberen Ecke dargestellt. Optional ist es auch möglich, ein Icon zur Unterscheidung von partiellen und totalen Termangaben zu verwenden.

Xcerpt-Konstrukte wie *desc* (descendant) und *all*, und *variablen* werden als Boxen in verschiedenen Grautönen dargestellt und sind oft durch zusätzliche textuelle ,,Verzierungen” gekennzeichnet. Variablen werden durch schwarze Boxen visualisiert, die den Variablennamen in weißer Schrift enthalten. Wenn eine

Variable zusäätzlich durch einen Anfrageterm beschränkt ist (durch das $\rightsquigarrow$ Konstrukt), dann erscheint dieser Term innerhalb der Variablenbox.

Regeln werden durch einen Pfeil visualisiert, der den Regelaspekt betont. Der Konstruktionsteil befindet sich links des Pfeils, der Anfrageteil rechts des Pfeils (siehe Abbildung 1).

Dynamische Eigenschaften. Klicken auf den Reiter einer Box *faltet* die zugehörige Box ein oder aus. VisXcerpt-Programme können durch Copy-and-Paste Befehle bearbeitet werden, die durch Kontextmenüs erreichbar sind. Um neue Elemente anzulegen, wird ein eigenes Dokument zur Verfügung gestellt, das alle visXcerpt-Konstrukte enthält. Wird der Mauszeiger auf eine Variable bewegt, so werden alle Vorkommen dieser Variable hervorgehoben. Referenzen in Termen werden durch Hyperlinks repräsentiert.

Danksagung

Diese Arbeit wurde im Rahmen des Projekts REWERSE (Nr 506779, `http://www.rewerse.net`) im 6th Framework Programme teilweise gefördert durch die Europäische Kommission und das Schweizerische Bundesamt für Bildung und Wissenschaft

Literatur

[1] Bry, F., Schaffert, S.: A Gentle Introduction into Xcerpt, a Rule-based Query and Transformation Language for XML. In: Proc. Int. Workshop on Rule Markup Languages for Business Rules on the Semantic Web. (2002) (invited article).

[2] Ceri, S., Damiani, E., Fraternali, P., Paraboschi, S., Tanca, L.: XML-GL: A Graphical Language for Querying and Restructuring XML Documents. In: Sistemi Evoluti per Basi di Dati. (1999)

[3] Consens, M., Mendelzon, A.: Expressing Structural Hypertext Queries in GraphLog. In: Second ACM Hypertext Conf. (1989) 269–292

[4] Pietriga, E., Quint, V., Vion-Dury, J.Y.: VXT: A Visual Approach to XML Transformations. In: ACM Symp. on Document Engineering. (2001)

[5] Munroe, K.D., Papakonstantinou, Y.: BBQ: A Visual Interface for Integrated Browsing and Querying of XML. In: VDB. (2000)

[6] Erwig, M.: A Visual Language for XML. In: IEEE Symp. on Visual Languages. (2000) 47–54

[7] Buneman, P., Fernandez, M., Suciu, D.: UnQL: A Query Language and Algebra for Semistructured Data Based on Structural Recursion. VLDB Journal **9** (2000) 76–110

[8] Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D.: A Query Language for XML. In: Proc. of Eighth Int. WWW Conf. (1999)

[9] Bry, F., Schaffert, S.: Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. In: Proc. Int. Conf. on Logic Programming (ICLP). LNCS 2401, Springer-Verlag (2002)

[10] Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web. From Relations to Semistructured Data and XML. Morgan Kaufmann (2000)