

Seminar: Spezifikations- und Selektionsmethoden für Daten und Dienste

XQuery Semantik

Yukari Ishii

Universität Karlsruhe (TH)
Institut für Programmstrukturen und Datenorganisation (IPD)

1 Einleitung

Die formale Semantik für XQuery 1.0 und XPath 2.0 wurde von der XML Query Working Group und der XSL Working Group definiert und stellt eine Ergänzung zu den XQuery 1.0 und XPath 2.0 Spezifikationen dar. Das Ziel der formalen Semantik ist die Definition der Bedeutung von XPath/XQuery Ausdrücken mit mathematischer Genauigkeit. Dies wird erreicht durch die Verwendung formaler Notationen zur Beschreibung von XPath/XQuery Objekten (z.B. Ausdrücken oder XML Schema Typen) und durch die systematische Definition der Beziehungen zwischen diesen Objekten. Für die Beschreibung von Manipulationen auf Objekten werden Grammatikproduktionen verwendet. Die formale Semantik für XQuery soll die Bedeutung der Sprache XQuery klar festlegen und alle Spezialfälle abdecken. Ferner soll sie als Referenz für die Implementierung dienen.

2 Verarbeitung von XQuery Anfragen

Die interne Verarbeitung von XQuery Anfragen orientiert sich an einem logischen Verarbeitungsmodell. Dieses ist nicht als feste Vorgabe für die Implementierung zu verstehen, jedoch sollte jede Implementierung der formalen Spezifikation so erfolgen, dass das Resultat der Auswertung einer XQuery Anfrage dem Resultat entspricht, was man bei Durchlaufen des logischen Modells erhalten hätte. Das logische Verarbeitungsmodell unterscheidet 2 Phasen, die **statische Analyse (static analysis)** und die **dynamische Auswertung (dynamic evaluation)**. Diese Phasen bestehen wiederum aus mehreren Teilschritten, so dass sich folgende Gliederung ergibt:

- Phase 1 **Static Analysis**:
 1. Parsing
 2. Static Context Processing
 3. Normalization
 4. Static Type Analysis
- Phase 2 **Dynamic Evaluation**
 1. Dynamic Context Processing
 2. Dynamic Evaluation

Die genannten Verarbeitungsphasen betreffen ausschließlich die interne Verarbeitung des XPath/XQuery-Prozessors und befassen sich nicht mit Aspekten der externen Interaktion, insbesondere auch nicht mit dem Zugriff auf Dokumente. Daher unterstützt eine typische XPath/XQuery Engine weitere Phasen, beispielsweise für die Verarbeitung eines importierten Schemas, die jedoch nicht Teil der formalen Semantik-Spezifikation sind.

Im Folgenden soll die Verarbeitung von XQuery-Anfragen in Anlehnung an das logische 2-Phasenmodell beschrieben werden.

2.1 Grundlagen

Ein Grundbaustein der formalen Spezifikation ist das sogenannte Judgment. Judgments sind Aussagen, die entweder gelten oder nicht gelten.

Beispiel 1. Das Judgment

$Expr \Rightarrow Value$

ist zu lesen als Ausdruck $Expr$ ergibt Wert $Value$. Es gilt, wenn die Auswertung des Ausdrucks $Expr$ den Wert $Value$ ergibt.

Beispiel 2. Das Judgment

$Expr : Type$

ist zu lesen als Ausdruck $Expr$ ist vom Typ $Type$. Es gilt, wenn der Ausdruck $Expr$ vom Typ $Type$ ist.

Die logische Beziehung zwischen verschiedenen Judgments lässt sich mit Hilfe von Inferenzregeln (inference rules) beschreiben. Eine Inferenzregel kann als eine Sammlung von 0 oder mehr Prämissen und einer Folgerung aufgefasst werden. Sowohl die Prämissen als auch die Folgerung sind Judgments. Eine Inferenzregel beschreibt, wie ein Complex Judgment aus einfacheren Premise Judgments gefolgert werden kann. Dabei wird die folgende Notation verwendet:

$$\frac{\text{Prämisse}[1] \dots \text{Prämisse}[n]}{\text{Folgerung}}$$

Die Notation ist so zu verstehen, dass wenn alle Prämissen gelten, auch die Folgerung gilt.

Beispiel 3.

$$\frac{\$x \Rightarrow 0 \quad 3 \Rightarrow 3}{\$x + 3 \Rightarrow 3}$$

Eine Inferenzregel ohne Prämissen gilt immer.

Beispiel 4.

$$\frac{}{3 \Rightarrow 3}$$

Logische Inferenzregeln nutzen Umgebungen (environments), um Informationen aus Berechnungen, die während der statischen Analyse und dynamischen Auswertung ausgeführt werden, abzulegen, so dass diese dann von anderen logischen Inferenzregeln genutzt werden können. Eine Umgebung kann dabei als ein Wörterbuch aufgefasst werden, welches ein Symbol, wie z. B. einen Funktionsnamen oder Variablennamen, auf ein Objekt, z. B. einen Funktionsrumpf, Typ oder

Wert, abbildet. Die in einer Umgebung gespeicherte Information kann sowohl nur abgerufen als auch aktualisiert werden. Umgebungen werden zu Gruppen zusammengefaßt, wobei in der formalen Semantik hauptsächlich zwei Gruppen von Bedeutung sind: Die dynamische Umgebung **dynEnv** und die statische Umgebung **statEnv**.

2.2 Statische Analyse (Static Analysis)

Die statische Semantik setzt XQuery Ausdrücke in Bezug zu dem zugrundegelegten XML Schema Typ. Ziel der statischen Analyse ist es, bestimmte Fehler wie beispielsweise Syntaxfehler oder Typfehler bereits zur Compilierungszeit anstatt erst zur Laufzeit zu erkennen. Dabei hängt diese Verarbeitungsphase nur von dem betrachteten Ausdruck selbst und dessen statischen Kontext ab. Sie verwendet, abgesehen von Schemata, keine anderen Eingabedaten. Tritt während der Verarbeitung kein Fehler auf, so ist das Ergebnis der statischen Analyse eine compilierte Form des Ausdrucks/der Anfrage. Die statische Analysephase umfaßt 4 Teilschritte, die im Folgenden erläutert werden.

Parsing Der Teilschritt parsing wird in der Spezifikation der formalen Semantik nicht näher beschrieben. Die formale Semantik definiert kein formales Modell für Syntaxbäume, sondern verwendet die XPath/XQuery Syntax. Tritt kein Syntaxfehler auf, so wird für den Ausdruck ein interner Operationsbaum generiert.

Static Context Processing Unter dem Ausdruckskontext eines gegebenen Ausdrucks versteht man all diejenigen Informationen, die das Resultat der Verarbeitung dieses Ausdrucks beeinflussen können. Diese Informationen werden unterteilt in den statischen und den dynamischen Kontext und werden entsprechend durch die Umgebungsgruppen *statEnv* und *dynEnv* erfaßt. Die Gruppe *statEnv* faßt alle Umgebungen zusammen, welche während der statischen Analyse zur Verfügung stehen. Ein Beispiel für ein Element aus dieser Gruppe ist die Umgebung mit dem Namen function declaration environment (*statEnv.funcType*), welche die statischen Typsignaturen von Funktionen speichert. Da XPath/XQuery mehrere Funktionen mit demselben Funktionsnamen zulassen, die sich nur in der Anzahl und der Signatur der Funktionsparameter unterscheiden, kann mit Hilfe dieser Umgebung eine Zuordnung zu der passenden Funktionsdeklaration erfolgen. Die statische Semantik von Ausdrücken und Anfragen ist abhängig vom statischen Eingabekontext. Dieser muß daher vor der Analyse des Ausdrucks/der Anfrage generiert werden.

Normalization Um die Spezifikation der Semantik zu vereinfachen, werden die gegebenen Ausdrücke/Anfragen durch eine einfachere aber redundante Darstellung ersetzt. So kann beispielsweise ein komplexer for-Ausdruck in einer FLWR-Anfrage durch eine Verschachtelung von mehreren einfachen for-Ausdrücken ersetzt werden. Die Sprache, die aus diesen einfachen Ausdrücken besteht, wird als

XPath/XQuery Core bezeichnet und durch eine Grammatik beschrieben, welche eine Untermenge der XQuery Grammatik ist. Während der Normalisierung wird jeder Ausdruck/jede Anfrage auf eine dazu äquivalente XPath/XQuery Core-Darstellung abgebildet. Dies erfolgt durch die Anwendung von Abbildungsregeln (mapping rules), die beschreiben, wie ein gegebener XPath/XQuery-Ausdruck als XPath/XQuery Core-Ausdruck geschrieben wird. Die Notation erfolgt in der Form:

```
statEnv |- [Object] [Subscript] == Mapped Object
```

Dabei steht links vom Symbol == das Originalobjekt gefolgt vom *Subscript*, welches die Art des Objektes angibt und darüberhinaus weitere Informationen zum Austausch zwischen verschiedenen Abbildungsregeln enthalten kann. Auf der rechten Seite wird die Ersetzung angegeben. Da das Vorhandensein des statischen Kontextes für die Normalisierung implizit vorausgesetzt wird, kann die Angabe *statEnv* auch weggelassen werden. Die Kurzform lautet dann:

```
[Object] [Subscript]
      ==
Mapped Object
```

Static Type Analysis Der Teilschritt static type analysis ist optional. Wird er nicht unterstützt, so folgt auf die Normalisierung die dynamische Auswertung. Die static type analysis prüft für jeden Ausdruck/jede Anfrage, ob sie typsicher ist. Ist dies der Fall, wird der statische Typ ermittelt.

Die Analyse erfolgt durch die Anwendung von sogenannten Typinferenzregeln (type inference rules) auf die Ausdrücke. Typinferenzregeln setzen XPath/XQuery Ausdrücke zu Typen in Bezug und spezifizieren, unter welchen Bedingungen ein Ausdruck wohltypisiert ist. Es muß sichergestellt werden, dass sich für jede mögliche Auswertung des Ausdrucks ein gültiger Wert ergibt. Dies wird mit dem nachfolgenden Beispiel veranschaulicht.

Beispiel 5. Gegeben sei der Ausdruck

```
let $v := 3 return $v+5
```

Dieser Ausdruck hat den Typ xs:integer, da

- die Eingabeliterale 3 und 5 Ganzzahlen sind
- die Eingabevariable \$v ebenfalls vom Typ Ganzzahl ist
- und die Summe zweier Ganzzahlen wieder eine Ganzzahl ergibt

Typinferenzregeln werden bottom-up auf Ausdrücke angewendet. Somit hängt der statische Typ eines Ausdrucks allein von den Typen seiner Teilausdrücke ab. Der statische Typalgorithmus wird rekursiv auf die Teilausdrücke angewendet, wobei in jedem Rekursionsschritt eine passende Typinferenzregel gefunden werden muß. Ansonsten ergibt sich ein Typfehler. In der statischen Analyse können

nur bestimmte Arten von Fehlern erkannt werden, andere sind erst während der Auswertung erkennbar. So kann hier eine unzulässige Vergleichsoperation, z. B. zwischen einer Ganzzahl und einer Zeichenkette, erkannt werden. Hingegen ließe sich ein Überlauf bei einer arithmetischen Operation auf 32-bit Ganzzahlen `xs:int` erst während der Auswertung erkennen. Die static type analysis ist nur für Core-Ausdrücke definiert. Bei erfolgreichem Abschluß der static type analysis wird ein abstrakter Syntaxbaum ausgegeben, in dem jeder Teilausdruck eine statische Typannotation besitzt.

Abschließend soll noch ein Beispiel für eine statische Typinferenzregel betrachtet werden.

```
Beispiel 6. statEnv |- Expr[1] : xs:boolean
              statEnv |- Expr[2] : Type[2]
              statEnv |- Expr[3] : Type[3]
```

```
-----
statEnv |- if Expr[1] then Expr[2] else Expr[3]
          : ( Type[2] | Type[3] )
```

Diese Regel besagt: Wenn der Bedingungsausdruck eines if-Ausdrucks vom Typ `boolean` ist, so entspricht der Typ des gesamten Ausdrucks einem der beiden Typen der `then`- oder der `else`-Klausel. Dabei wird der Ergebnistyp als Vereinigung angegeben.

2.3 Dynamische Auswertung (Dynamic Evaluation)

Die dynamische Auswertungsphase (dynamic evaluation phase, gelegentlich auch als *execution* bezeichnet) wertet eine Anfrage auf Eingabedokument(en) aus. Die dynamische Semantik setzt einen XQuery Ausdruck in Bezug zu dem Wert, der sich bei dessen Auswertung ergibt. Diese Verarbeitungsphase umfaßt zwei Teilschritte, auf die im Folgenden kurz eingegangen werden soll.

Dynamic Context Processing Die dynamische Semantik von Ausdrücken/Anfragen ist abhängig vom dynamischen Eingabekontext. Dieser muß daher vor der eigentlichen Auswertung generiert werden. Der dynamische Kontext wird durch die Umgebungsgruppe *dynEnv* erfaßt. Diese enthält beispielsweise die Umgebung *date-time* (*dynEnv.dateTime*), welche dem aktuellen Datum und der aktuellen Uhrzeit im dynamischen XPath/XQuery Kontext entspricht. Für die dynamische Auswertung wird sowohl die Gruppe *statEnv* als auch die Gruppe *dynEnv* benötigt. Die statische Umgebung wird beispielsweise gebraucht für Typzuordnungen während der Auswertung.

Dynamic Evaluation In diesem Teilschritt wird nun konkret der Wert eines Ausdrucks/einer Anfrage berechnet. Dazu werden Auswertungsregeln bottom-up auf gegebene Ausdrücke angewendet, angefangen bei den Literalen und Variablen. Dieses Prinzip wird an dem Beispiel aus der static type analysis veranschaulicht.

Beispiel 7. Die Auswertung des Ausdrucks

```
let $v := 3 return $v+5
```

ergibt 8. Dies wird wie folgt bottom-up hergeleitet:

- Die Literale 3 und 5 haben als Wert 3 bzw. 5
- die Variable \$v hat den Wert 3
- $3+5=8$

Die Auswertungssemantik ist nur für Core-Ausdrücke/Anfragen definiert. Sie wird mit Hilfe von Wertinferenzregeln (value inference rules) spezifiziert, welche Ausdrücke zu Werten in Bezug setzen und in manchen Fällen auch die Auswertungsreihenfolge eines Ausdrucks festlegen. Die Auswertung liefert einen Wert oder einen dynamischen Fehler. Ein dynamischer Fehler ergibt sich beispielsweise dann, wenn die Auswertung eines Ausdrucks von einer Komponente des dynamischen Kontextes abhängt, für welche kein Wert festgelegt ist.

3 Zusammenfassung

Die formale Semantik für XQuery 1.0 und XPath 2.0 definiert die Bedeutung von XPath/XQuery Ausdrücken und ist somit eine Ergänzung zur XQuery 1.0 bzw. XPath 2.0 Spezifikation. Als Orientierung dient ein logisches Verarbeitungsmodell, welches aus zwei Hauptphasen besteht, die ihrerseits in weitere Phasen unterteilt sind. Implementierungen müssen nicht der vorgegebenen Einteilung folgen. Phasen können überlappt oder sogar parallel ausgeführt werden. Wichtig ist nur, dass die Herleitung des Resultates konform zur Spezifikation ist.

Literatur

1. W3C: XQuery 1.0 and XPath 2.0 Formal Semantics, W3C Working Draft vom 11. Feb. 2005, <http://www.w3.org/TR/2005/WD-xquery-semantics-20050211/>